Guía fácil

AMSTRAD



I . Ramón
P. Buera
V. Trigo

PARANINFO S.A.

Guía fácil del

AMSTRAD

I. RAMON
P. BUERA
V. TRIGO



1987

PARANINFO S.A.

MADRID

Impreso en España
Printed in Spain

ISBN: 84-283-1519-1

Depósito Legal: M-4458-1987

PARANINFO S.A. Magallanes, 25 - 28015 MADRID (02301/37/27)

ALCO, artes gráficas. Jaspe, 34 - 28026 MADRID

Indice de materias

1
El teclado

Como creemos que la conexión y la puesta en marcha del ordenador no ofrecerá dificultad alguna para el lector, en este primer apartado comenzaremos con la descripción del teclado.

Una vez conectado el ordenador y a continuación del mensaje de bienvenida:

```
Amstrad 128k Microcomputer (s3)

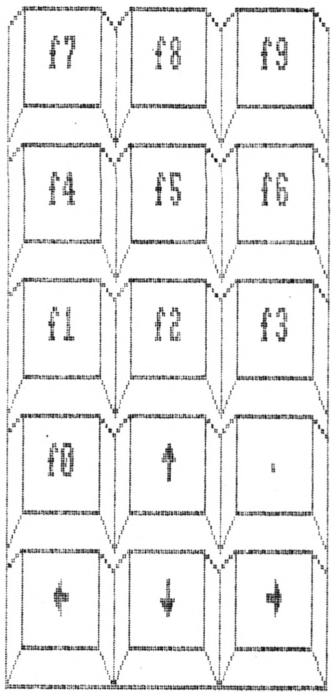
©1985 Amstrad Consumer Electronics plc
          and Locomotive Software Ltd.

BASIC 1.1

READY
█
```

se observará la presencia de un cuadradito del tamaño de un carácter que llamaremos "CURSOR" y que nos indica la posición del siguiente carácter a imprimir en la pantalla.

En la parte derecha del teclado están situadas las teclas que componen el "TECLADO NUMERICO" y las teclas de "MOVIMIENTO DEL CURSOR". Con estas últimas podremos desplazar dicho cursor en sentido horizontal o vertical por toda la pantalla.



Teclado numérico (CPC 6128)

Las teclas de la parte izquierda del teclado tienen una distribución semejante a las de una máquina de escribir y pueden clasificarse en tres grupos:

- *Teclas alfabéticas,* que llevan marcado un único carácter alfabético (A, B, C,.......)
- *Teclas marcadas con dos caracteres,* que podrán ser numéricos o especiales. Por ejemplo:

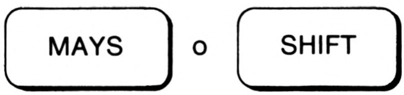


– *Teclas de control,* que van marcadas con varios caracteres alfabéticos y que analizaremos a continuación.

TECLAS DE CONTROL

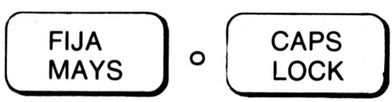
Pueden recibir diferentes denominaciones según se trate del teclado en versión española o en versión inglesa. Inicialmente daremos ambas denominaciones pero más adelante sólo emplearemos la primera.

Describiremos a continuación la misión de alguna de estas teclas:



Se observará enseguida que al presionar una tecla marcada con un carácter alfabético, éste aparecerá, en minúsculas, en la pantalla. Sin embargo, si simultáneamente se presiona la tecla "MAYS" aparecerá dicho carácter pero esta vez en mayúsculas.

Al pulsar una de las teclas que llevan impresos dos caracteres, el que está situado en la parte inferior es el que aparece en la pantalla; si se pulsa simultáneamente con la tecla "MAYS" se obtiene el carácter superior.



Presionando en una ocasión esta tecla se consigue escribir con mayúsculas permanentemente hasta que se pulse de nuevo para volver a minúsculas.

No tiene ningún efecto sobre las teclas con dos caracteres. Con esto se quiere decir que es el carácter inferior el que aparece en la pantalla se haya presionado

FIJA MAYS

o no. Para conseguir el carácter superior hay que pulsar simultáneamente

MAYS

Si queremos que sean los caracteres superiores los que aparezcan al pulsar una de esta teclas, deberemos pulsar simultáneamente las teclas

FIJA MAYS y CONTROL

Para anular este efecto se deben volver a pulsar dichas teclas.

BORR o DEL

Con esta tecla se consigue borrar el carácter situado a la izquierda del cursor.

CLR

Cuando situamos el cursor sobre un carácter escrito en la pantalla, éste aparece en inverso. Con la tecla

CLR

se puede borrar dicho carácter.

Ahora veremos, con un ejemplo, la manera de usar estas dos teclas que nos ayudan a corregir errores. Supongamos que hemos escrito:

Paradisipar algunass dydas

en lugar de:

Para disipar algunas dudas

y ahora nos toca enmendar los errores que hemos cometido.

Empezaremos corrigiendo de izquierda a derecha, por lo que situaremos el cursor sobre la letra "d" de disipar y pulsaremos la barra espaciadora para separar las dos palabras con un espacio.

Para borrar una de las dos "eses" puedo emplear dos procedimientos:

1) Colocar el cursor en cualquiera de las dos letras y pulsar la tecla

CLR

2) Colocar el cursor sobre la segunda "ese" y pulsar la tecla

BORR

Y, para acabar, se borra la letra "y" por cualquiera de los dos procedimientos anteriores y se pulsa la tecla "u". Quedando, de esta forma, corregidos todos los errores.

Las teclas de control:

COPIA o COPY

INTRO o ENTER

CONTROL o CTRL

así como:

RETURN

que aparecen en el modelo CPC 6128, se verán en los siguientes capítulos.

NOTAS

1- Al completar una línea de caracteres en la pantalla, el cursor salta a la línea siguiente automáticamente.

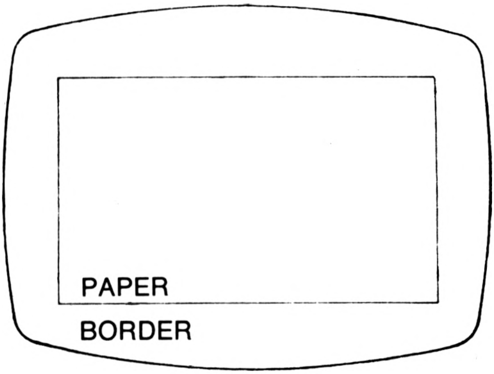
2- En la pantalla del ordenador se distingue el número cero "0" y la vocal "O".

3- En lugar de la coma decimal se empleará el punto decimal. Así en lugar de 3,14,pondremos 3.14 para que el ordenador lo entienda.

4- También se puede escribir .3 en lugar de 0.3 suprimiendo el cero a la izquierda del punto decimal.

2
La Pantalla

Ya sabemos que nuestro ordenador lleva consigo una pantalla para visualizar toda la información que se indique en los diferentes programas que hagamos. Esta pantalla está dividida en dos zonas:



De estas dos zonas, la única disponible para imprimir es PAPER. Veamos como hacerlo.

En primer lugar debemos saber que PAPER está dividida internamente en filas y columnas, configurando una cuadrícula en la que cada cuadro será el espacio reservado para escribir un carácter.

Una de las características más llamativas de nuestro ordenador es que admite tres tipos distintos de cuadrícula; es decir, podemos escribir con letras de tres tamaños diferentes. Estos tres tipos de cuadrícula vienen definidos por la instrucción:

MODE

Esta orden lleva asignado un número (0, 1 ó 2) que determina la cuadrícula de acuerdo con el siguiente cuadro:

	nº filas	nº columnas
MODE 0	25	20
MODE 1	25	40
MODE 2	25	80

Como se observa en el cuadro, la única variación que existirá entre los tres tamaños posibles será su anchura.

Para ver su efecto imprimiremos una frase en la pantalla. Esto se conseguirá con la orden.

PRINT

Así, si introduces el programa

```
10 MODE 0
20 PRINT"AMSTRAD"
```

al ejecutarlo (con RUN, como ya sabes) aparecerá

```
AMSTRAD
Ready
█
```

Si ahora sustituyes la línea 10, escribiéndola de nuevo, por

```
10 MODE 1
```

obtendrás

```
AMSTRAD
Ready
█
```

Finalmente, con

```
10 MODE 2
```

aparece en pantalla

```
AMSTRAD
Ready
█
```

Veamos algunas características de estas órdenes que es interesante conocer:

- La instrucción MODE, como habrás comprobado, borra la pantalla.
- Al conectar el ordenador, se trabaja, mientras no se ordene otra cosa, en MODE 1.
- Desgraciadamente no podemos utilizar más de un MODE simultáneamente.

- La instrucción PRINT puede sustituirse por su abreviatura ?, lo cual nos permite evitar errores al teclear y ganar tiempo.
- Las comillas de la derecha que encierran la frase en la orden PRINT, pueden suprimirse si así se desea.

CORRECCION DE ERRORES

Es posible que hayas cometido algún error al teclear cualquiera de las instrucciones anteriores, y te hayas visto obligado a desconectar el aparato o a teclear la línea de nuevo. Hay dos procedimientos para corregir estos errores sin recurrir a métodos tan drásticos.

Supongamos que has escrito

```
10 MODEO
20 PRIMT "AMSTRAD"
```

que, como observarás, tienen tres errores:

- En la línea 10 aparece la letra O en lugar de 0.
- No existe separación entre MODE y el número que debe acompañar a esta instrucción.
- En la línea 20 está escrito PRIMT en lugar de PRINT.

El primero de los procedimientos es el llamado de **método de edición.** Consiste en editar la línea a modificar y con las teclas de desplazamiento del cursor efectuar las correcciones oportunas.

Para editar la línea correspondiente utilizamos la orden

EDIT n

donde n es el número de línea.

En nuestro caso, para corregir la línea 10, teclearemos

EDIT 10

con lo cual se presenta en pantalla una copia de dicha línea con el cursor incluido. Después desplazaremos el cursor hasta situarlo encima de la letra O y, pulsando la barra espaciadora, obtendremos el espacio que nos hacía falta.

A continuación, para sustituir la letra O por el número 0, existen dos posibilidades:

- Colocar el cursor a la derecha de la letra O y pulsar la tecla

  ←BORR

  De este modo conseguiremos borrarla. Posteriormente escribiremos en su lugar 0.

- Colocar el cursor sobre la letra O y con la tecla

  CLR

  hacer desaparecer dicha letra; ya estamos en disposición de teclear el símbolo correcto.

En ambos casos, una vez subsanado el error, basta pulsar

INTRO

para que la línea corregida sustituya a la errónea.

El segundo de los procedimientos de corrección es el llamado **método del cursor de copia**. Este consiste en la utilización de un segundo cursor que se obtiene pulsando la tecla.

MAYS

simultáneamente con una de las cuatro teclas de movimiento del cursor.

En primer lugar debemos colocar este cursor al comienzo de la línea a modificar, en nuestro caso la línea 20. Cada vez que pulsemos la tecla

COPIA

obtendremos la repetición de un carácter de dicha línea. En nuestro caso la pulsaremos seis veces con el fin de llegar a la letra M. A continuación, se escribe directamente la letra N. Ahora tenemos que impedir que, al pulsar la tecla

COPIA

se imprima la letra M. Esto se consigue presionando la tecla

MAYS junto con →

Por último, seguimos copiando (con COPIA) hasta el final de la línea para después, con

INTRO

introducir esta nueva línea en la memoria.

SEPARADORES

Como ya sabes, en BASIC, los dos separadores habituales son:

- COMA ,
- PUNTO Y COMA ;

En nuestro ordenador su efecto depende del MODE en que trabajamos y del tipo de variable a separar.

Así, el separador ; (punto y coma) presenta las expresiones alfanuméricas una junto a la otra sin dejar separación entre ellas. Por ejemplo

```
PRINT "GUIA;"AMSTRAD
```

imprime

```
GUIAAMSTRAD
```

En cambio, cuando trabajamos con expresiones numéricas, se reserva un espacio a la derecha y otro a la izquierda para su posible signo. Por ejemplo

```
PRINT 12;-1;2
```

imprime

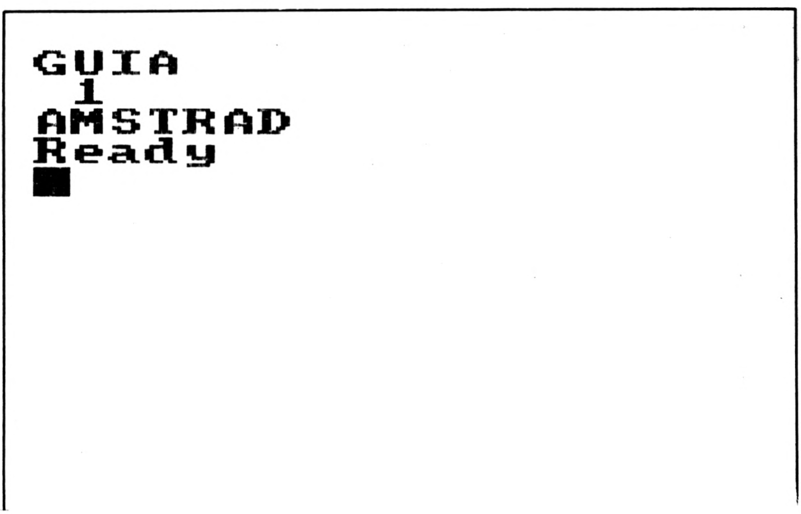
```
12 -1  2
```

El efecto del separador , (coma) depende del modo en el que se trabaje, además de la puntualización hecha anteriormente para el caso de expresiones numéricas.

Con MODE 0 se salta a la línea siguiente. Así, al ejecutar

```
10 MODE 0
20 PRINT"GUIA",1,"AMSTRAD"
```

se obtiene



En MODE 1 el efecto del separador , (coma) es saltar un tercio de línea. Por ejemplo, con

```
10 MODE 1
20 PRINT"GUIA",1,-2,3,"AMSTRAD"
```

aparece en pantalla

```
GUIA          1             -2
 3            AMSTRAD
Ready
█
```

Por último, con MODE 2, cada vez que aparece una coma se produce un salto igual a un sexto de línea. Podemos comprobarlo introduciendo

```
10 MODE 2
20 PRINT"GUIA",1,2,3,4,"AMSTRAD",5
```

y viendo que se obtiene

```
GUIA       1         2         3         4         AMSTRAD
 5
Ready
█
```

Resumiendo, el efecto preestablecido del separador coma es:

MODO	SALTO
0	1 línea
1	1/3 línea
2	1/6 línea

Nuestro AMSTRAD nos ofrece la posibilidad de variar estos saltos empleando la orden

ZONE n

donde n indicará el número de columnas que saltará al encontrar el separador , (coma).

Por ejemplo, con:

```
10 MODE 1
20 ZONE 8
30 PRINT 1,-2,3,-4,5,-6,7
```

se obtiene

```
 1      -2      3      -4      5
-6       7
```

Todavía, nuestro ordenador, nos ofrece una nueva posibilidad de efectuar separaciones utilizando la orden

SPC n

que se encargará de dejar n espacios en blanco.

Podemos comprobarlo con

```
10 MODE 1
20 PRINT"AMSTRAD"
30 PRINT 1;SPC(4);-2
```

que, al ejecutarse, mostrará en pantalla

```
AMSTRAD
 1     -2
```

Con esta orden debemos tener en cuenta lo siguiente:

- El separador ; (punto y coma) puede omitirse.
- Si n es un número decimal positivo, se redondea al entero más próximo; y si es negativo, se toma como cero.
- El máximo número de espacios que puede conseguirse con SPC es 19, 39 ó 79 según que el modo de trabajo sea respectivamente 0, 1 ó 2.

LOCALIZACION EN PANTALLA

En este apartado conoceremos tres instrucciones (LOCATE, TAB y USING) que nos permitirán imprimir en cualquier sitio de la pantalla. De esta forma, lograremos mejorar ampliamente la presentación visual de nuestros datos y resultados.

Con la orden

LOCATE c,f

se coloca el cursor de texto en la posición indicada por la columna c y la figura f.

Las columnas se cuentan de izquierda a derecha (de 1 a 20, 40 u 80 según el modo de trabajo) y las filas de arriba abajo (de 1 a 25).

Así, por ejemplo, con

```
10 MODE 1
20 LOCATE 17,13
30 PRINT"AMSTRAD"
```

imprimimos el nombre de nuestro ordenador en el centro de la pantalla.

Se suele utilizar a menudo esta orden para desplazar el mensaje READY a la última fila, con el fin de que no estorbe la presentación de nuestros trabajos.

Si el mensaje a imprimir no tiene suficiente espacio en la línea indicada por LOCATE, o bien el número de columna es mayor que el máximo permitido por el modo de trabajo, se imprime la totalidad de la expresión a partir de la primera columna de la fila siguiente.

Como es fácil suponer, los parámetros c y f que acompañan a LOCATE deben ser positivos. Si son decimales se redondearán al entero más próximo.

Mientras la instrucción LOCATE fija la fila y la columna a partir de las cuales se empieza a escribir, TAB fija únicamente la columna. Esta segunda instrucción será especialmente útil cuando deseemos presentar tablas ordenadas en columnas. Por ejemplo, una tabla que muestra los precios de diferentes artículos podría hacerse de la siguiente forma:

```
10 MODE 1
20 PRINT
30 PRINT TAB(5);"ARTICULO";TAB(17);"PREC
IO";TAB(29);"EXISTENCIAS"
40 PRINT TAB(5);"--------";TAB(17);"----
--";TAB(29);"-----------"
```

```
50 PRINT TAB(6);"QUESO";TAB(17);1600;TAB
(31);112;"Kg"
60 PRINT TAB(6);"ACEITE";TAB(17);250;TAB
(31);1300;"l"
70 PRINT TAB(6);"VINO";TAB(17);85;TAB(31
);630;"l"
80 PRINT TAB(6);"JAMON";TAB(17);1250;TAB
(31);65;"pz"
90 LOCATE 1,24
```

En pantalla aparecerá

```
ARTICULO        PRECIO          EXISTENCIAS
--------        ------          -----------
 QUESO           1600               112 Kg
 ACEITE          250                1300 l
 VINO            85                 630 l
 JAMON           1250               65 pz




Ready
█
```

Como habrás observado, las cantidades numéricas correspondientes a las tablas "PRECIO" y "EXISTENCIAS" se han ajustado por la izquierda. Esto no es lo habitual en tablas de este tipo, ya que lo más normal es efectuar el ajuste por la derecha. Nuestro ordenador también ofrece esta posibilidad usando la instrucción.

USING

cuya forma general es

PRINT USING "# #.....#";K

que reserva tantos espacios como símbolos # para la impresión de K. Si éste ocupa menos espacios, se completa con espacios en blanco situados a su izquierda. Así, la tabla anterior se puede presentar en su forma más corriente

```
ARTICULO       PRECIO         EXISTENCIAS
--------       ------         -----------
 QUESO          1600              112 Kg
 ACEITE          250             1300 l
 VINO             85              630 l
 JAMON          1250               65 pz





Ready
█
```

con el programa

```
10 MODE 1
20 PRINT
30 PRINT TAB(5);"ARTICULO";TAB(17);"PREC
IO";TAB(29);"EXISTENCIAS"
40 PRINT TAB(5);"--------";TAB(17);"----
--";TAB(29);"-----------"
50 PRINT TAB(6);"QUESO";TAB(17);USING"##
###";1600;:PRINT TAB(31);USING "#####";1
12;:PRINT" Kg"
```

```
60 PRINT TAB(6);"ACEITE";TAB(17);USING "
#####";250;:PRINT TAB(31);USING "#####";
1300;:PRINT " 1"
70 PRINT TAB(6);"VINO";TAB(17);USING "##
###";85;:PRINT TAB(31);USING "#####";630
;:PRINT" 1"
80  PRINT TAB(6);"JAMON";TAB(17);USING"#
####";1250;:PRINT TAB(31);USING"#####";6
5;:PRINT" pz"
90 LOCATE 1,24
```

VENTANAS

La introducción y salida de datos se efectúa en nuestro ordenador a través de 10 canales diferentes numerados del 0 al 9. Los ocho primeros canales, numerados del 0 al 7, presentan el texto en pantalla, mientras que los dos restantes están asignados a la impresora (canal 8) y al disco o casete (canal 9).

Para designar un canal hay que colocar antes de su número, el símbolo #. Así con

PRINT # 8,"AMSTRAD

se obtiene por impresora la palabra AMSTRAD.

Los ocho primeros canales de presentación en pantalla tienen asignada una ventana que inicialmente ocupa toda la pantalla. Las dimensiones de estas ventanas pueden alterarse según nuestro interés con la instrucción

WINDOW

cuya expresión general es:

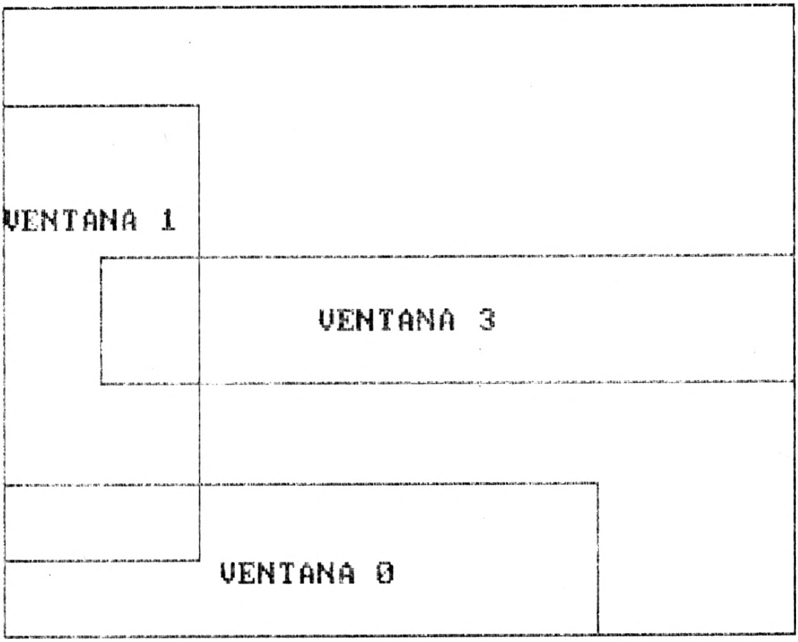
WINDOW # n,c_1,c_2,f_1,f_2

donde n es el número de la ventana (comprendido entre 0 y 7) y c_1,c_2,f_1 y f_2 son las dos columnas (c_1,c_2) y las dos filas (f_1,f_2) que delimitan la ventana. Si no se indica el número de ventana n, el ordenador considera que se trata de la ventana número 0, que es además la que utiliza siempre para emitir sus mensajes.

Con el siguiente programa redefinimos tres ventanas

```
10 MODE 1
20 WINDOW 1,30,20,25
30 WINDOW#1, 1,10,5,22
40 WINDOW#3, 5,40,10,15
```

que ocuparán en pantalla las zonas



Las restantes ventanas seguirán ocupando la totalidad de la pantalla.

Cuando en una instrucción hagamos referencia a una ventana determinada, será necesario consignar siempre su número precedido del símbolo #. Así, si quisiéramos imprimir el número de cada una de las ventanas definidas anteriormente en el centro de dichas ventanas, bastaría con añadir:

```
50 LOCATE 12,4:PRINT"VENTANA 0"
60 LOCATE#1,1,5:PRINT#1,"VENTANA 1"
70 LOCATE#3,13,4:PRINT#3,"VENTANA 3"
```

Como puedes observar, la fila y la columna asociadas a la instrucción LOCATE empiezan a contabilizarse a partir del extremo superior izquierdo de la ventana correspondiente.

3
Continuando con el Basic

COMANDOS

Todos los comandos habituales en BASIC están también en nuestro ordenador. Demos un rápido repaso a éstos indicando alguna de sus particularidades.

– RUN

Es la orden encargada de ejecutar los programas. Admite la posibilidad de escribir

RUN X

con lo que se ejecutaría el programa a partir de la línea número X.

A diferencia de otros ordenadores, esta orden no borra la pantalla.

– CLS

Esta instrucción tiene como misión borrar la pantalla, aunque el programa permanece en memoria en su totalidad. Además el modo de trabajo se sigue manteniendo.

– CLEAR

Con CLEAR conseguimos que los valores asignados a las diferentes variables desaparezcan.

– NEW

Al ejecutarse esta orden desaparece el programa alojado en ese momento en la memoria, manteniéndose el mismo modo de trabajo y no borrándose la pantalla.

Como ya sabes, al desconectar el ordenador, los programas también desaparecen. El mismo efecto se consigue pulsando simultáneamente las tres teclas siguientes:

CONTROL | MAYS | ESC

Además de estos comandos, nuestro ordenador posee otros que serán de gran utilidad a la hora de manejar los listados de los programas. Son los siguientes:

– LIST p–q

Presenta en pantalla las líneas comprendidas entre la línea número p y la línea número q ambas inclusive. Esta instrucción admite otras modalidades según falten los parámetros p ó q

LIST p–	Lista desde la línea p hasta al final
LIST –q	Lista desde el principio hasta la línea q
LIST p	Lista solamente la línea número p
LIST	Lista todo el programa

Cuando el programa es largo es posible que, si deseamos listarlo en su totalidad, no quepa en pantalla. En ese caso se produce un efecto de SCROLL que consiste en que desaparece la línea superior para aparecer una nueva por la parte inferior de la pantalla. En muchos casos será necesario detener este efecto para visualizar el listado más cómodamente. Se consigue una detención momentánea pulsando la tecla

ESC

Si a continuación pulsamos cualquier otra tecla, continúa el efecto SCROLL. Sin embargo, si se pulsa de nuevo esta tecla, la detención del efecto SCROLL es definitiva.

Al listar un programa puede observarse que las líneas se numeran de menor a mayor independientemente del orden de introducción. Además, todas las instrucciones aparecerán en mayúsculas excepto en el caso de que exista algún error.

– DELETE p–q

Suprime de la memoria todas las líneas del programa comprendidas entre la línea número p y la línea número q ambas inclusive.

Como la orden anterior, también admite otras modalidades.

DELETE p–	Borra desde la línea p hasta el final
DELETE –q	Borra desde el principio hasta la línea q
DELETE p	Borra solamente la línea número p
DELETE	Borra todo el programa

– AUTO p,q

Coloca de forma automática los números de línea del programa a medida que dichas líneas se introducen. De esta forma conseguimos evitar el teclearlos.

Los dos parámetros asociados a AUTO sirven para indicar el número de la primera línea (p) y la separación entre dos números de líneas consecutivas (q).

También esta instrucción admite otras posibles formas. Así:

AUTO p	Marca el número de la primera línea con p y las siguientes las separa de 10 en 10.

AUTO, q	Asigna a la primera línea el número 10 y da un incremento de q unidades a la numeración de las siguientes.
AUTO	Empieza en la línea número 10 y da un incremento de 10 unidades para las siguientes.

Cuando hayamos introducido la totalidad del programa pulsaremos la tecla ESC para dejar sin efecto a la instrucción AUTO.

– RENUN p,q,r

Con esta orden logramos reenumerar los números de línea de un programa ya introducido. El parámetro p indica el nuevo número de la primera línea a reenumerar; q es su antiguo número y r es el incremento que otorgaremos, dentro de la nueva numeración, a dos líneas consecutivas cualesquiera.

Si faltan en la instrucción RENUM p ó r, se consideran como 10; mientras que si falta q, se considera que la reenumeración se efectúa a partir de la primera línea del programa.

OPERADORES NUMERICOS

Nuestro ordenador, además de los operadores aritméticos habituales

+	SUMA
–	RESTA
*	PRODUCTO
/	DIVISION
↑	POTENCIACION

posee otros dos que facilitan enormemente el cálculo numérico:

MOD	RESTO ENTERO
\	COCIENTE ENTERO

Estos dos nuevos operadores tiene como misión la obtención del cociente y el resto de una división entera. Así con

a MOD b

obtenemos el resto resultante al efectuar la división de a entre b; mientras que con

a \ b

se conseguiría el cociente de dicha división. Por ejemplo

49 MOD 8 = 1

49 \ 8 = 6

Cuando trabajamos con varios operadores tenemos que tener presente el orden de prioridad de éstos para conocer el proceso que seguirá nuestro ordenador al efectuar las operaciones.

Este orden de prioridad es:

1° ↑

2° MOD

3° * y /

4° \

5° + y −

que, como ya es sabido, puede alterarse con el uso de paréntesis.

Para practicar un poco puedes completar el siguiente cuadro sin utilizar el ordenador.

OPERACION	RESULTADO
2 + 3 * 5	17
(2 + 3) * 5	25
8 MOD 3 * 5	
2 * 5 \ 6/3	
2 ↑ (2 * 3) + 14 * MOD (9 \ 4)	
2 ↑ 2 * 3 + 14 * MOD 9 \ 4	

VARIABLES

El AMSTRAD admite tres tipos de variables. A saber:

- Variable numérica real
- Variable numérica entera
- Variable alfanumérica

La *variable numérica real* puede tomar cualquier valor real comprendido entre

$$\pm 2.9 * 10^{-39} \text{ y } \pm 1.7 * 10^{38}$$

Para denotar este tipo de variables utilizamos una letra seguida de dígitos o letras. Es decir, serán variables numéricas reales las designadas por:

A
ab
aB1

La *variable numérica entera* sólo puede tomar valores enteros comprendidos entre los números

– 32768 y 32767

Este tipo de variables se designarán como las anteriores añadiéndoles a continuación el símbolo %. Por ejemplo, serán variables enteras

A%
ab%
aB1%

Siempre que trabajemos con números enteros del intervalo anterior será conveniente emplear variables enteras con el fin de ganar rapidez de ejecución y ahorrar memoria.

La *variable alfanumérica* está formada por caracteres de los tipos alfabéticos, numéricos o especiales encerrados entre comillas. Por ejemplo,

```
"PEPE"
"A1 - a"
"1234"
```

Estas variables se designan como las reales añadiendo al final el símbolo $. Por ejemplo,

```
A$
ab$
aB1$
```

Como ya se ha indicado anteriormente, las comillas que cierran una variable alfanumérica pueden eliminarse. También hay que tener en cuenta que el máximo número de caracteres que puede llevar una variable alfanumérica es de 255.

Para dar nombre a una variable puede utilizarse la instrucción

```
LET
```

Así con

```
LET a = 5
```

asociamos a la variable real de nombre a, el número 5.

Nuestro ordenador nos permite definir una variable sin tener que teclear LET. Por tanto, serían correctas las asignaciones siguientes:

```
a = 5
A1% = 236
Ba$ = "CARLOS
```

Al trabajar con variables debemos tener presente que no podemos designar a ninguna de ellas con los nombres reservados a las instrucciones BASIC. Por ejemplo, es incorrecta la asignación

```
SQR = 23
```

Por otra parte, como sucede en todos los ordenadores, cuando el número real tiene más de nueve dígitos, éste se expresa utilizando la notación exponencial. Según esto

```
1234567890       se expresa    1.23456789 E +09
- 0.0000003789   se expresa    - 3.789 E -07
```

Finalmente, es conveniente conocer que cuando una variable numérica no está definida, se le asocia automáticamente el valor 0. De la misma forma, una variable alfanumérica no definida se considera como la cadena vacía.

Con el fin de estructurar mejor nuestros programas podemos reservar unas ciertas letras del abecedario para cada uno de los tipos de variables. Esto se consigue con las órdenes

DEFINT

DEFREAL

DEFSTR

Cada una de ellas tiene tres posibles modalidades. Así, por ejemplo, en el primer caso

DEFINT a	Considera como variables enteras todas aquellas cuyo nombre comience por la letra a sin necesidad de escribir el símbolo %.
DEFINT a,b,c	Considera como variables enteras todas aquellas cuyo nombre comience por las letras a, b ó c.
DEFINT b–m	Considera como variables enteras todas aquellas cuyo nombre comience por una letra comprendida entre b y m ambas inclusive.

Las instrucciones DEFREAL y DEFSTR funcionan de manera análoga a DEFINT para los casos de variable numérica real y variable alfanumérica respectivamente.

INTRODUCCION DE DATOS DESDE EL TECLADO

Ya hemos visto en el apartado anterior que podemos introducir datos en el listado de un programa con la orden LET, aunque no sea preciso escribirla. Sin embargo, en la mayoría de los programas es necesario introducir datos desde el teclado. La orden:

INPUT

es la encargada de efectuar este trabajo.

Para comprender el funcionamiento de esta orden veamos el siguiente programa:

```
10 INPUT a
20 INPUT b
30 PRINT a*b
```

que nos pide dos números y escribe su producto.

Al ejecutarse el programa con RUN, aparece un símbolo ? para indicarnos que el ordenador espera que tecleemos un número al cual le asignará el nombre a. De forma análoga debemos introducir otro número asociado a la variable b.

Con el fin de facilitar el manejo del programa, podemos añadir a continuación de INPUT, un mensaje aclaratorio, sobre la variable a introducir. Por ejemplo, el programa anterior podría escribirse:

```
10 INPUT "PRIMER NUMERO";a
20 INPUT "SEGUNDO NUMERO";b
30 PRINT"SU PRODUCTO ES";a*b
```

Si en INPUT utilizamos el separador punto y coma (;) entre el mensaje y el nombre de la variable, aparece, al ejecutarse el programa, el símbolo ?. En cambio, no aparecerá este símbolo, si el separador es una coma (,).

Con el fin de reducir la extensión de nuestros programas, en un mismo INPUT podemos introducir varios datos, siempre que se separen por comas. Así, las líneas 10 y 20 anteriores podrían sustituirse por:

```
10 INPUT "ESCRIBE DOS NUMEROS";a,b
```

En este caso habrá que introducir los dos datos separados por una coma.

Para dominar esta importante instrucción sería conveniente que construyeras unos sencillos programas, para resolver problemas como éstos:

- Pasar de dólares a pesetas y viceversa.
- Calcular el área de un triángulo.
- Calcular el resto y el cociente de una división entera.
- Calcular la velocidad media conocido el espacio y el tiempo.

También con INPUT podemos introducir variables alfanuméricas de forma similar. En este caso no será necesario escribir las comillas que encierran toda variable alfanumérica.

Al introducir variables alfanuméricas, no podemos, por ahora, escribir comas dentro de ellas, ya que, según hemos visto anteriormente, el ordenador consideraría a éstas como separadores de diversas variables. Si queremos introducir comas dentro de una variable, es necesario utilizar la orden:

LINE INPUT

Así, por ejemplo, si al ejecutar el programa:

```
10 LINE INPUT A$
20 PRINT A$
```

tecleamos:

LLEGUE, VI, VENCI

se observará que en pantalla, aparece correctamente escrita (línea 20) la frase introducida.

En cambio podríamos comprobar que si sustituimos la línea 10 por:

```
10 INPUT A$
```

aparecerá el mensaje de error:

Redo from start

al introducir la frase anterior.

4
Saltos

Hasta ahora, en los pocos programas que se han visto, éstos se han ejecutado según el orden señalado por su número de línea. En muchos casos este orden debe alterarse para conseguir lo que nos proponemos. La orden:

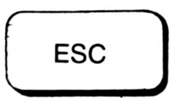
GOTO n

traslada el control de ejecución del programa a la línea número n del programa.

Por ejemplo, el programa:

```
10 PRINT"AMSTRAD"
20 GOTO 10
```

imprime indefinidamente la palabra AMSTRAD, ya que cada vez que el programa llega a la línea número 20, se devuelve el control a la línea número 10. Este programa estará funcionando continuamente hasta que lo paremos desde el teclado con



Otro sencillo programa que utiliza esta instrucción es el que obtiene la impresión de los sucesivos números naturales.

```
10 PRINT A
20 A=A+1
30 GOTO 10
```

Observa que, en la línea 10, al no estar definida inicialmente la variable A, se le asigna el valor 0. Por otro lado, es interesante destacar la línea 20, ya que aparece una expresión que no tiene significado matemático, pero sí informático. En ella asignamos a A su valor anterior más una unidad. De esta forma conseguiremos que A tome todos los valores naturales.

Como ejercicio puedes intentar obtener los programas que impriman:

- Todos los números pares.
- Todos los números impares.
- Todos los números múltiplos de 3.
- Todos los pares mayores que 100.

En BASIC existe otra modalidad de SALTO en la cual se recuerda la línea desde la cual se ha efectuado. La orden que realiza estos saltos con retorno es:

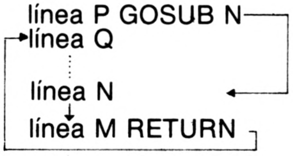
GOSUB n

Su funcionamiento es el siguiente: al alcanzar la línea donde está esta orden, se transfiere el control del programa a la línea n. A partir de ésta, se ejecuta el programa de una forma secuencial hasta encontrar la orden:

RETURN

En ese momento el control del programa se devuelve a la orden siguiente a GOSUB n.

Con el siguiente esquema se comprenderá mejor su funcionamiento:



La parte del programa comprendida entre las líneas N y M, ambas inclusive, recibe el nombre de *Subrutina.*

Estas subrutinas se suelen emplear cuando deseamos que una parte del programa se repita varias veces a lo largo de su ejecución.

Por ejemplo, el siguiente programa nos pide nombres de cinco letras y los escribe subrayados en la pantalla.

```
10 INPUT "PRIMER NOMBRE";A$
20 INPUT "SEGUNDO NOMBRE";B$
30 INPUT "TERCER NOMBRE";C$
40 CLS
50 F=5:N$=A$:GOSUB 100
60 F=10:N$=B$:GOSUB 100
70 F=15:N$=C$:GOSUB 100
80 GOTO 80
100 LOCATE 17,F:PRINT N$
110 LOCATE 17,F+1:PRINT"====="
120 RETURN
```

La subrutina, en este caso, está formada por las líneas 100, 110 y 120. Su misión es imprimir los nombres (N$) y después subrayarlos.

Es necesario colocar la línea 80 con el fin de que el programa, una vez ejecutada la línea 70, no se introduzca en la subrutina; ya que, si esto se produjera, se imprimiría el mensaje de error:

Unexpected RETURN in 120

Otra solución sería obligar al programa a que se detuviera en la línea 80. Esto se consigue con las órdenes:

STOP o END

La diferencia entre estas dos instrucciones consiste en que si la detención del programa se debe a STOP, podemos continuar la ejecución del programa tecleando la orden: CONT.

En cambio si es END la causa de la detención, CONT no tiene efecto sobre ella.

SALTOS MULTIPLES

A menudo el salto a efectuar depende del posible valor que pueda tomar una variable determinada. En estos casos existe la orden:

ON GOTO

para solucionar este problema:

Su función general es:

ON A GOTO P,Q,R,S,...

donde P,Q,R,S,... deben ser números de líneas del programa.

Con esta orden se produce un salto a la línea número P, cuando A sea igual a 1. Si A vale 2, el salto se realiza a la línea Q y así sucesivamente.

Un programa que ayuda a comprender cómo funciona esta instrucción es el siguiente:

```
10 INPUT "NUMERO DEL 1 AL 7";A
20 ON A GOTO 40,50,60,70,80,90,100
30 END
40 PRINT"LUNES":STOP
50 PRINT"MARTES":STOP
60 PRINT"MIERCOLES":STOP
70 PRINT"JUEVES":STOP
80 PRINT"VIERNES":STOP
90 PRINT"SABADO":STOP
100 PRINT"DOMINGO":STOP
```

En este programa se nos pide (línea 10) un número comprendido entre 1 y 7 y posteriormente se imprime el día de la semana correspondiente a dicho número.

Como ejercicio puedes hacer los dos programas siguientes:

- Dado el número de un mes, escribir su nombre.
- Dado un número del 0 al 9, escribir su nombre.

También existe una orden análoga para subrutinas:

ON GOSUB.

Su forma general y funcionamiento es similar a ON GOTO, con la diferencia de que ahora el salto es con retorno.

Estos dos saltos múltiples se emplean muy a menudo en todos aquellos programas donde figura un menú de opciones.

RELACIONES

Los seis operadores de relación que nos permitirán establecer condiciones, son los siguientes:

=	(igual)
>	(mayor que)
<	(menor que)
>=	(mayor o igual que)
<=	(menor o igual que)
<>	(distinto)

Así, por ejemplo, si queremos expresar que una variable X sea mayor o igual que 3, escribiremos:

X >= 3

En muchos casos es necesaria más de una condición. Supongamos que queremos expresar que una variable numérica está comprendida entre 1 y 6. Con sólo los operadores vistos anteriormente, es imposible conseguirlo, ya que, en este caso, debemos operar con dos condiciones simultáneamente. Para solucionar estos problemas existen lo que llamamos *operadores lógicos*. Con ellos logramos unir dos o más condiciones en una sola. Estos operadores son:

AND	(conjunción lógica)
OR	(disyunción lógica)
XOR	(disyunción exclusiva)
NOT	(negación lógica)

Si C_1 y C_2 son dos condiciones cualesquiera, el funcionamiento de cada uno de estos operadores lógicos es el siguiente:

C_1 AND C_2 — es una nueva condición que será cierta cuando lo sean a la vez C_1 y C_2 En cualquier otro caso, será falsa.

C_1 OR C_2	es una condición que se considerá cierta cuando una de las dos lo sea. Será falsa pues cuando ambas lo sean.
C_1 XOR C_2	será falsa cuando las dos sean ciertas o las dos falsas. En caso contrario, es cierta.
NOT C_1	será cierta cuando C_1 sea falsa y viceversa.

Así, el caso anteriormente expuesto, puede solucionarse tecleando:

A>1 AND A<6

De la misma manera que sucedía con los operadores aritméticos, también existe una orden de prioridad entre estos operadores lógicos:

1°	NOT
2°	AND
3°	OR
4°	XOR

Este orden de prioridad podemos alterarlo con el uso de paréntesis. Así si:

X=−1   Y=6   Z=28

la condición:

X<−2 AND Y<1 OR Z>14

sería cierta, mientras que:

X<−2 AND (Y<1 OR Z>14)

sería falsa.

CONDICIONALES

Intentemos resolver un sencillo problema:

"Dado un número entero, el ordenador debe decir si es par o impar".

Para resolver éste problema será necesario tener alguna instrucción que nos permita expresar la siguiente idea:

Si el número es divisible por dos, entonces escribir PAR.
Si el número no es divisible por dos, escribir IMPAR.

Esta instrucción condicional (si.... entonces....) es:

IF... THEN...

siendo su estructura general:

IF condición THEN acción a ejecutar.

Cuando el ordenador encuentra esta instrucción, analiza si la condición es cierta o falsa. En el primero de los casos ejecuta la acción o acciones que hay tras THEN. En caso contrario, condición falsa, se pasa a la línea siguiente.

Por tanto, el problema que se planteaba al principio, puede resolverse con el siguiente programa:

```
10 INPUT "ESCRIBE UN NUMERO";N
20  IF N MOD 2=0 THEN PRINT"PAR"
30 IF N MOD 2<>0 THEN PRINT"IMPAR"
```

Hasta ahora habíamos visto cómo se podían realizar saltos contínuos (sin fin) con la instrucción GOTO. Si junto a esta instrucción manejamos la instrucción IF, podemos conseguir saltos determinados (se repiten un número finito de veces). Por ejemplo, el siguiente programa imprime 20 veces la palabra AMSTRAD :

```
10 PRINT"AMSTRAD"
20 C=C+1
30 IF C=20 THEN STOP
40 GOTO 10
```

Recuerda que la variable C aumenta una unidad cada vez que se ejecuta la línea 20. De la misma forma puedes resolver los siguientes problemas:

- Escribir los 10 primeros números.
- Escribir los 20 primeros pares.
- Escribir los pares de cuatro cifras.
- Calcular la suma de los 100 primeros números naturales.
- Calcular la suma de los impares de tres cifras.
- Calcular el producto de los 15 primeros números.
- Dado un número, escribir su signo.
- Dadas 20 cantidades, calcular su media.
- Dibujar con asteriscos un cuadrado centrado en la pantalla.
- Dibujar con asteriscos un conjunto de rectas paralelas verticales.
- Dado un número, escribir su tabla de multiplicar.

Cuando la orden a ejecutar tras THEN es un salto a la línea n, podemos utilizar cualquiera de las tres instrucciones siguientes:

IF condición THEN GOTO n

IF condición THEN n

IF condición GOTO n

Con el fin de simplificar y estructurar mejor nuestros programas, a veces utilizaremos la orden:

ELSE

acompañando a IF... THEN...

Su forma general es:

IF condición THEN acciones 1 ELSE acciones 2.

Cuando la condición es cierta, se ejecutan las acciones 1. En caso contrario, se realizan las acciones 2. Utilizando esta instrucción, el programa que pedía la clasificación en par o impar de un número, puede quedar del siguiente modo:

```
10 INPUT "ESCRIBE EL NUMERO";N
20 IF N MOD 2=0 THEN PRINT"PAR" ELSE PRINT"IMPAR"
```

5
Bucles

Cuando dentro de un programa se repite un mismo proceso varias veces, es más cómodo y dosificador utilizar lo que llamamos un bucle FOR–NEXT, que un condicional junto con un contador.

La forma general de este bucle es:

```
FOR contador = inicio TO fin
     .
     .
     .
     proceso a repetir
     .
     .
     .
NEXT contador.
```

Así, el programa del apartado anterior, que imprimía 20 veces la palabra "AMSTRAD" puede hacerse del siguiente modo:

```
10 FOR C=1 TO 20
20 PRINT"AMSTRAD"
30 NEXT C
```

La variable C tomará inicialmente el valor 1. Cada vez que el control del programa llegue a la instrucción NEXT, aumenta en una unidad su valor, devolviéndose el control a la línea 20.

Cuando el valor de C es superior a 20 finaliza la ejecución del bucle.

Los bucles FOR–NEXT tienen en nuestro ordenador unas características que facilitarán nuestros trabajos. A saber:

- El nombre del contador puede constar de más de un carácter.
- En la orden NEXT podemos omitir el nombre del contador.
- Siempre que sea posible será conveniente utilizar variables enteras.
- Podemos salir de un bucle con la instrucción GOTO; sin embargo, no es posible entrar dentro de él.

En muchos programas el contador debe incrementarse no en una unidad, sino en varias. Podemos conseguir este efecto con la instrucción:

STEP

colocada a continuación del valor final del contador:

FOR contador = inicio TO fin STEP incremento

Así, si queremos escribir todos los múltiplos de 3 menores que 100, teclearemos el siguiente programa:

```
10 FOR X=3 TO 100 STEP 3
20 PRINT X
30 NEXT
```

El incremento del contador puede ser también negativo o decimal.

Muchos de los programas propuestos en el apartado anterior puedes hacerlos utilizando bucles. Además, te proponemos algún otro:

- Escribir los números de 1 al 100 en orden inverso.
- Sumar todos los números comprendidos entre dos números cualesquiera.
- Calcular la suma de los cuadrados de los números pares de tres cifras.
- Escribir todos los cuadrados perfectos menores que 1000.
- Dado un número ver si es primo o no.

- Con asteriscos dibujar un enrejado.
- Con asteriscos dibujar un cuadrado y sus diagonales.
- Introducir 20 números y que el ordenador indique el mayor y el menor de ellos.

BUCLES ENCAJADOS

Cuando en el proceso a repetir dentro de un bucle aparece a su vez otro bucle, obtenemos la estructura denominada *bucles encajados.*

Veamos el siguiente programa:

"Cinco caballos numerados del 1 al 5 participan en una carrera. Expresar todos los posibles órdenes de llegada para el primer y segundo clasificado".

```
10 MODE 1
20 PRINT"PRIMERO","SEGUNDO":PRINT
30 FOR X=1 TO 5
40 FOR Y=1 TO 5
50 IF X<>Y THEN PRINT X,Y
60 NEXT Y
70 NEXT X
```

La tabla de posibles resultados que se presentará en pantalla será la que aparece en la página siguiente.

Para cada valor del contador X se realizará todo el bucle 40–60.

Como ya se ha dicho anteriormente, nuestro AMSTRAD nos permite no escribir los nombres de los contadores en las líneas 60 y 70. Además existen otras posibilidades, pudiéndose sustituir estas líneas por cualquiera de las tres siguientes:

```
60 NEXT:NEXT

60 NEXT Y:NEXT X

60 NEXT Y,X
```

También tenemos que tener presente que, al trabajar con bucles encajados, para que éstos funcionen perfectamente, los bucles interiores deben estar totalmente abarcados por los exteriores.

Te proponemos más ejercicios para que te sirvan de entretenimiento y distracción:

```
PRIMERO          SEGUNDO

 1                2
 1                3
 1                4
 1                5
 2                1
 2                3
 2                4
 2                5
 3                1
 3                2
 3                4
 3                5
 4                1
 4                2
 4                3
 4                5
 5                1
 5                2
 5                3
 5                4
Ready
█
```

- Escribir la numeración de las 28 cifras del dominó.
- Escribir todos los posibles resultados obtenidos al lanzar tres dados.
- Escribir los números del 1 al 49 según el modelo de la Lotería Primitiva.
- Presentar centrado en la pantalla, un reloj digital, que funcione aproximadamente.
- Hacer un programa que encuentre todos los números de tres cifras que sea igual a la suma de los cubos de las tres cifras que lo forman. (Por ejemplo 153).
- Hacer un programa que permita construir, hasta la fila deseada, el triángulo de Floyd:

      1
      2 3
      4 5 6
      7 8 9 10
      ...
      ...

BUCLES CONDICIONADOS

Nuestro ordenador posee una nueva estructura de bucle (WHILE–WEND) que en algunos casos, presenta ventajas respecto a los bucles FOR–NEXT.

En estos bucles, el proceso se repite mientras se cumple una determinada condición.

Su expresión general es:

```
WHILE condición
        .
        .
        .
        proceso a repetir
        .
        .
        .
WEND
```

Supongamos que queremos calcular la media de las estaturas de un número indeterminado de personas. Este problema puede resolverse con el siguiente programa:

```
10 E=1
20 WHILE E>0
30 INPUT "ESTATURA";E
40 S=S+E:C=C+1
50 WEND
60 PRINT"MEDIA";(S-E)/(C-1)
```

De acuerdo con la línea 20 bastará introducir como estatura un valor 0 ó negativo para finalizar el bucle. En la línea 60 se ha colocado la expresión (S–E)/(C–1) con el fin de anular el último dato por no ser válido.

Respecto a estos bucles podemos indicar las siguientes características:

- Si la condición es falsa inicialmente, el bucle no se ejecuta.
- Evidentemente, esta condición puede constar de varias condiciones unidas por operadores lógicos.
- De forma análoga a los bucles FOR–NEXT, también podemos encajar bucles WHILE–WEND.

6
Lectura de datos

Muchas veces nos encontramos en un programa con una gran cantidad de datos fijos que debemos almacenar. Según hemos visto anteriormente, esto podemos conseguirlo con las órdenes LET o INPUT. Sin embargo, este procedimiento acostumbra a ser excesivamente reiterativo, lo cual conlleva un trabajo excesivo y un aumento de la probabilidad de cometer errores. Estos problemas pueden solucionarse con las instrucciones:

READ (leer)

DATA (datos)

Con la orden DATA lograremos almacenar los datos; mientras que con la orden READ se leerán asignándoles un nombre.

Un ejemplo sencillo que permite comprender la utilidad de estas dos instrucciones es la confección de un breve listín telefónico.

```
10 FOR X=1 TO 5
20 READ A,B$
30 PRINT B$,A
40 NEXT
50 DATA 401050,PEDRO,321586,VICENTE,6305
76,IGNACIO,430150,PILAR,156318,CARMEN
```

Al ejecutarse la línea 20 por primera vez, se asigna a las variables A y B$ la primera pareja de datos almacenados en la instrucción

DATA de la línea 50. Al repetirse por segunda vez el bucle, se les asigna la siguiente pareja de datos, y así sucesivamente.

El funcionamiento de estas dos instrucciones (READ, DATA) está sujeto a una serie de reglas que es conveniente conocer para sacarles el máximo partido:

- La orden DATA puede colocarse en cualquier sitio del programa, pero es conveniente que se sitúe al principio o al final con el fin de clarificar y estructurar nuestros programas.
- El número de variables a leer nunca puede ser superior al número de datos almacenados. Si este número de datos es tan elevado que no cabe en una sola instrucción DATA, siempre podemos habilitar nuevas líneas DATA, de tal forma que, al acabar de tomar los valores consignados en la primera línea DATA, se pasaría a considerar los de la siguiente línea.
- También puede leerse un dato concreto del conjunto que tenemos almacenado en un DATA con un programa como el siguiente:

```
10 INPUT "DATO BUSCADO";N
20 FOR X=1 TO N
30 READ A
40 NEXT
50 PRINT"EL DATO";N;"ES";A
60 DATA 2,4,6,8,10,12,14,16,18,20
```

Basándote en todo lo visto anteriormente, puedes resolver los siguientes ejercicios:

- Dado el número de un mes, escribir su nombre.
- Dado el número de un teléfono, escribir el nombre de su propietario o viceversa.
- Dado el nombre de una capital europea, escribir el país correspondiente o viceversa.
- Construir un breve diccionario inglés–francés–castellano de tal modo que, al introducir una palabra en un idioma, aparezca su traducción en los otros.

Damos la solución al primer ejercicio:

```
10 INPUT"NUMERO DEL MES";A
20 FOR X=1 TO A
30 READ B$
```

```
40 NEXT
50 PRINT"EL MES NUMERO";A;"ES ";B$
60 DATA "ENERO","FEBRERO","MARZO","ABRIL
"
70 DATA "MAYO","JUNIO","JULIO","AGOSTO"
80 DATA "SETIEMBRE",OCTUBRE","NOVIEMBRE"
,"DICIEMBRE"
```

Si con este programa deseásemos obtener los nombres de varios meses, no podríamos introducir la línea:

```
55 GOTO 10
```

ya que, al leer datos en la línea 30, continuaría a partir del mes impreso anteriormente.

Siempre nos quedaría la solución de no introducir esa línea número 55 y ejecutar con RUN el programa cada vez que deseemos un mes. Pero sería más cómodo tener una orden que permitiera restablecer los valores ya leídos en DATA. Esta orden es:

RESTORE

Con ella se comienza la lectura de datos desde el primer DATA.

Así, podríamos modificar el programa anterior, añadiendo las líneas:

```
53 RESTORE
55 GOTO 10
```

Esta orden tiene además otra posibilidad:

RESTORE n

donde n es el número de línea donde se encuentra el DATA a partir del cual queremos restablecer los datos.

7
Funciones numéricas

Para facilitar el cálculo de tipo científico, nuestro ordenador posee un conjunto de funciones numéricas estándar. Dichas funciones trabajan con variables numéricas y devuelven resultados también numéricos. A continuación expondremos estas funciones, indicando sus características principales:

– ABS (X)

Es la función valor absoluto; es decir, valdrá:

–X si X es negativo
X si X es positivo

– SGN (X)

Esta función depende del signo de X y su valor será:

–1 si X es negativo
0 si X es igual a 0
1 si X es positivo

– SQR (X)

Da la raiz cuadrada de X que, por tanto, deberá ser siempre positivo.

– EXP (X)

Calcula el valor e^X donde e=2.7182818...

Para que no presente mensaje de error X deberá ser menor que: 88.0296919

– LOG (X)

Es la función inversa de la anterior e indica el valor al cual hay que elevar **e** para obtener X. Este valor de X debe ser siempre positivo para que esta función tenga sentido.

– LOG10 (X)

Es una variante de la anterior función. Es el valor al cual hay que elevar 10 para obtener X.

– MAX (A,B,C...)

Da el valor máximo del conjunto de valores numéricos representados por A,B,C...

– MIN (A,B,C...)

Da el valor mínimo del conjunto de valores numéricos representados por A,B,C...

– SIN (X)

Calcula el valor del seno del ángulo X. Si no se indica lo contrario, este ángulo debe considerarse en radianes.

– COS (X)

Calcula el valor del coseno del ángulo X.

- TAN (X)

Calcula el valor de la tangente del ángulo X.

- ATN (X)

Es la función inversa de la anterior. Indica el ángulo en radianes, mientras no se indique lo contrario, cuya tangente es X.

- DEG

Con esta orden logramos que todos los ángulos se trabajen en grados sexagesimales en lugar de radianes.

- RAD

Es la orden contraria a la anterior y con ella logramos trabajar con radianes. Sólo será preciso utilizarla cuando hayamos introducido anteriormente DEG.

FUNCIONES ENTERAS

En este apartado veremos cuatro funciones que utilizaremos en gran número de ocasiones. Todas ellas dan como resultado un número entero, aunque existe una variante de una de ellas que dará lugar a números decimales.

Estas cuatro instrucciones serán:

- INT (X)

Obtiene el mayor de todos los números enteros menores o iguales que X. Es decir:

- Si X es positivo, INT (X) coincide con el número sin su parte decimal.
- Si X es negativo, INT (X) es el número sin su parte decimal menos una unidad.

Por ejemplo:

INT (3.8504)=3 INT (–3.8504)=–4

- Si el número X es entero INT (X) coincide con X. Precisamente esta particularidad se utilizará a menudo cuando queremos expresar que un número cualquiera debe ser entero.

– FIX (X)

Da el entero que resulta de suprimir la parte decimal de X. Por ejemplo:

FIX (3.8504)=3 FIX (–3.8504)=–3

Según las anteriores definiciones es evidente que:

- Si X es entero X=INT (X) =FIX (X)
- Si X es decimal positivo INT (X) =FIX (X)
- Si X es decimal negativo INT (X) =FIX (X)–1

– CINT (X)

Es el entero más cercano a X, siempre que esté comprendido entre –32768 y 32767. Por ejemplo:

CINT (3.8504)=4 CINT (–3.8504)=–4

– ROUND (X)

Es el entero más cercano a X, pero sin limitación para el valor de X.

Esta instrucción presenta una variante muy interesante a la hora de presentar resultados:

ROUND (X,N)

que redondea el número X con N decimales. Por ejemplo:

ROUND (3.8504,2)=3.85
ROUND (3.8504,1)=3.9

En el caso de que N sea igual a cero, se tiene:

ROUND (X,N) = ROUND (X)

Y si N es negativo se redondea a la potencia de 10^{-N} más cercana. Por ejemplo:

ROUND (4375.832, −1)=4380
ROUND (4375.832, −3)=4000

Puedes ejercitarte con estas nuevas instrucciones completando, sin ordenador, el siguiente cuadro:

VALOR DE X	INT (X)	FIX (X)	CINT (X)	ROUND (X)	ROUND (X,3)
4.835					
−2.563703					
163.5186					
−40356.370126					
230.457868					
−15.3765					
7800.0001					

8
Cadenas

Recordemos que una cadena o variable alfanumérica es un conjunto de caracteres de cualquiera de los tipos alfabéticos, numéricos o especiales. Recordemos también que su nombre debe finalizar con el símbolo $ y sus caracteres deben entrecomillarse. Por ejemplo:

```
A$= "AMSTRAD"
```

Entre cadenas existe una operación llamada suma o concatenación que obtiene la cadena que resulta de unir los caracteres de ambas cadenas. Así si:

```
A$= "MI"
B$="NOMBRE"
```

se tendría:

```
A$+B$= "MINOMBRE"
B$+A$= "NOMBREMI"
```

Existen tres funciones que relacionan las cadenas con números. Son las siguientes:

- LEN (A$)

Da el número total de caracteres que conforman la cadena A$, incluidos los espacios en blanco. Así, si:

A$ = "ME LLAMO JUAN"

se tiene que:

LEN (A$)= 13

- STR$ (X)

Devuelve la cadena formada por todos los caracteres de la variable numérica X, incluido el espacio reservado para el signo. Por ejemplo:

STR$ (25)= " 25"
STR$ (–25)="–25"

- VAL (A$)

Es la inversa de la anterior y devuelve el número formado por los caracteres de A$. Esta instrucción exige que los primeros caracteres de A$ sean dígitos, aunque pueden ir precedidos de signo o contener punto decimal. Por ejemplo:

VAL ("385")=385
VAL ("–4.32")=–4.32
VAL ("27AB48")=27

TROCEADO DE CADENAS

Dada una cadena, por ejemplo A$="AMSTRAD", podemos obtener subcadenas de ella aplicando las siguientes instrucciones:

- LEFT$ (A$,N)

Obtiene la subcadena formada por los N primeros caracteres de A$. Por ejemplo:

LEFT$ (A$,3)="AMS"
LEFT$ (A$,5)="AMSTR"

Si N es superior a la longitud de la cadena, considera la totalidad de ésta. Así:

LEFT$ (A$,9)="AMSTRAD"

- RIGHT$ (A$,N)

Es la subcadena formada por los N últimos caracteres de A$. Así:

RIGHT$ (A$,3)="RAD"
RIGHT$ (A$,5)="STRAD"
RIGHT$ (A$,9)="AMSTRAD"

- MID$ (A$,M,N)

Considera la subcadena formada por los N caracteres de A$ que se encuentran a la derecha del que ocupa la posición M, incluído éste. Por ejemplo:

MID$ (A$,3,2)="ST"
MID$ (A$,2,1)="M"

Si N es mayor que el número de caracteres disponibles, o si no se indica, se considera la subcadena formada por todos los caracteres que hay desde la posición M hasta el final. Por ejemplo:

MID$ (A$,5,7)="RAD"
MID$ (A$,6)="AD"

Veamos dos programas que utilizan estas instrucciones:

- Dado un nombre, escribirlo al revés.

```
10 INPUT "ESCRIBE UN NOMBRE";A$
20 L=LEN(A$)
30 FOR N=1 TO L
40 PRINT MID$(A$,L-N+1,1);
50 NEXT
```

– Dado un número entero positivo (con no más de 9 cifras) calcular la suma de todas sus cifras.

```
10 INPUT "ESCRIBE EL NUMERO";N
20 A$=MID$(STR$(N),2)
30 FOR X=1 TO LEN(A$)
40 S=S+VAL(MID$(A$,X,1))
50 NEXT
60 PRINT"LA SUMA ES";S
```

En la línea 20 hemos transformado el número en cadena desechando el primer carácter, por ser un espacio en blanco.

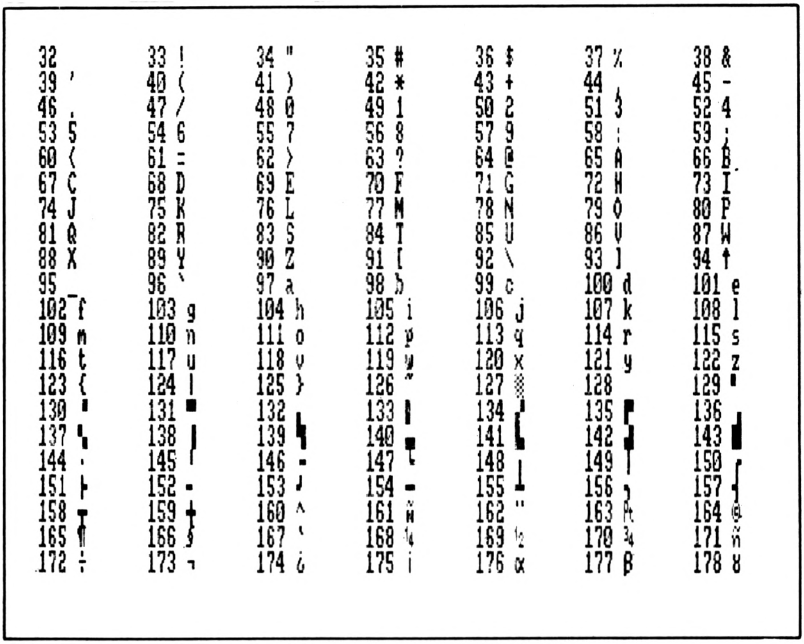
Para ejercitarte en el manejo de subcadenas puedes realizar los siguientes programas:

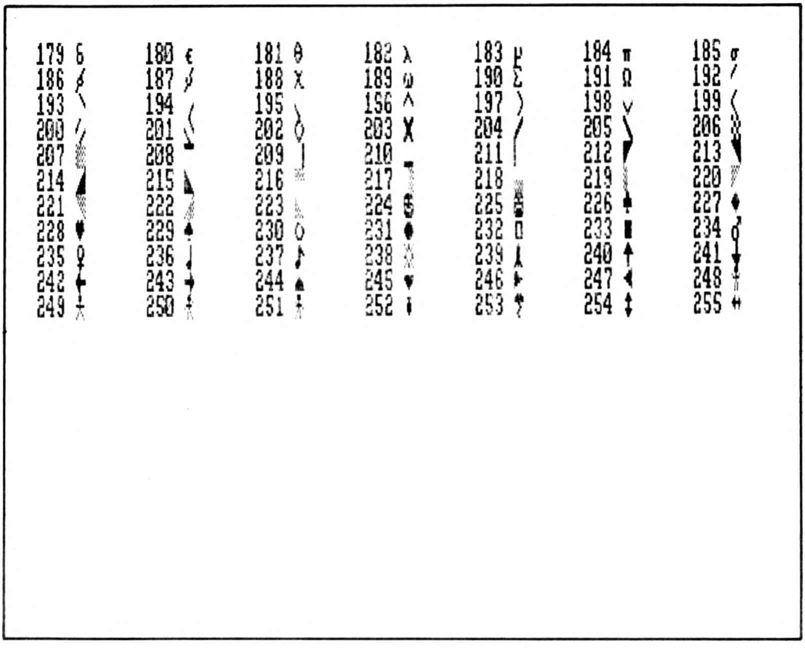
- Dado el infinitivo de un verbo, decir a qué conjugación pertenece.
- Dado el infinitivo de un verbo regular de la primera conjugación, escribir su presente de indicativo.
- Dada una frase, volverla a escribir sustituyendo los espacios en blanco por un guión.
- Dada una frase, sustituir todas sus vocales por asteriscos.
- Dada una frase, escribirla en vertical.
- Dada una frase, escribir las diferentes palabras que la conforman en líneas sucesivas.
- Dada una frase, contabilizar el número de vocales que aparecen en ella.
- Dado un número entero, decir si es o no capicúa.
- Dado un nombre, escribirlo centrado en la pantalla, independientemente del MODE en que se trabaje.

CODIGO ASCII

Todos los caracteres de nuestro ordenador tienen asignado un número que llamamos su código ASCII. Este código es un número entero comprendido entre 0 y 255. Los códigos comprendidos entre 0 y 31 corresponden a caracteres de control; los comprendidos entre 32 y 126 son comunes para casi todos los ordenadores, y los mayores de 126 son símbolos gráficos particulares de cada ordenador.

La siguiente tabla muestra los caracteres correspondientes a los códigos ASCII mayores que 31:





Existen dos órdenes que permiten trabajar con éstos códigos. Son las siguientes:

- ASC (A$)

Obtiene el código ASCII del primer carácter de la cadena A$. Así:

ASC ("JUAN")=74   ASC ("*↑38")=42

- CHR$ (N)

Devuelve el carácter cuyo código es N. Por ejemplo:

CHR$ (50)="2"   CHR$ (115)="s"

El siguiente programa imprime la tabla anterior:

```
10 MODE 2
20 ZONE 11
30 FOR x=32 TO 255
40 PRINT x;CHR$(x),
50 NEXT
```

Ahora puedes intentar escribir mensajes en clave. El sistema a seguir es muy sencillo: una vez dada la frase, se codifican cada uno de sus caracteres en el código ASCII sumándoles a todos los códigos anteriores un número prefijado, imprimiéndose la frase con los caracteres correspondientes a estos nuevos códigos. Para descifrar el mensaje se hará el proceso inverso, restando el número prefijado anteriormente a todos los códigos de los caracteres de la frase cifrada.

Como hemos dicho, los códigos que están comprendidos entre 0 y 31 se dedican a caracteres de control. Veamos alguno que puede resultarte útil:

PRINT CHR$ (7)	emite un pitido
PRINT CHR$ (8)	retrasa el cursor un lugar
PRINT CHR$ (9)	adelanta el cursor un lugar
PRINT CHR$ (10)	baja el cursor una línea manteniendo la columna.
PRINT CHR$ (24)	imprime en inverso

FUNCIONES DE CADENAS

Nuestro Amstrad posee una serie de funciones para trabajar con cadenas que, aunque no se usan muy a menudo, alguna vez puede ser interesante utilizar.

– LOWER$ (A$)

Devuelve la cadena formada por todos los caracteres de A$ sustituyendo las letras mayúsculas por sus correspondientes minúsculas. Por ejemplo:

LOWER$ ("Ab5C+") = "ab5c+"

– UPPER$ (A$)

Es la función contraria a la anterior. Sustituye las letras minúsculas de A$ por sus correspondientes mayúsculas. Así:

UPPER$ ("Ab5C+") = "AB5C+"

– SPACE$ (N)

Es la cadena formada por N espacios en blanco. N nunca puede ser mayor que 255.

– STRING$ (N,"carácter")

Es la cadena formada por N caracteres iguales al expresado en segundo lugar. Así:

STRING$ (5,"+") = "+++++"

En lugar de indicar el carácter, podemos expresar un código ASCII. Por tanto, la cadena anterior también se habría obtenido con:

STRING$ (5,43)

- INSTR (N,A$,B$)

Indica el número a partir de qué carácter de A$ está contenida la cadena B$. El parámetro N señala el carácter de A$ a partir del cual se comienza la búsqueda.

- BIN$ (X,Y)

Pasa a binario el número entero X. El parámetro Y indica el número de cifras binarias de X.

- HEX$ (X,Y)

Obtiene la cadena formada por el número X expresado en sistema hexadecimal, siendo Y el número de cifras necesarias.

9
Azar

En muchas situaciones: juegos, simulaciones, etc, interesa tener un método para obtener números al azar. Para resolver problemas de este tipo, los ordenadores llevan incorporada una lista de números aleatorios. En nuestra lista los números son decimales comprendidos entre 0 y 1.

La orden que permitirá sacar números de esta lista es:

```
RND(X)
```

que tiene tres versiones distintas según el signo de X.

Si X es positivo, al ejecutarse por primera vez RND (X) toma como valor el primer valor de la lista. Cuando esta orden vuelva a ejecutarse de nuevo, tomaría el siguiente número de la lista y así sucesivamente.

Por ejemplo, si queremos conseguir los veinte primeros números de esta lista, bastaría con realizar el siguiente programa, una vez conectado el aparato:

```
10 PRINT"LUGAR","NUMERO"
20 FOR X=1 TO 20
30 PRINT X,RND(1)
40 NEXT
```

Hemos hecho la observación de ejecutar el programa inmediatamente después de conectar el aparato, para evitar la posibilidad de que se haya manejado dicha función anteriormente.

Como, a fin de cuentas, lo que interesa en el parámetro que acompaña a RND es su signo, se hubiera conseguido el mismo efecto si en lugar de RND (1) se hubiera colocado en la línea 35: RND (5), RND (8), etc.

Compruébalo en el programa anterior, recordando que debes desconectar el aparato si quieres obtener los mismos números que la primera vez.

En el caso de X positivo, para ahorrarnos trabajo, nuestro ordenador permite no indicar X. Por tanto, en lugar de

RND (X)

podemos escribir:

RND

Si X es negativo, RND (X) toma siempre el mismo valor para cada X. Compruébalo con el siguiente programa:

```
10 FOR X=1 TO 20
20 PRINT RND(-5)
30 NEXT
```

Con este otro programa podrás obtener los valores aleatorios correspondientes a los 20 primeros números negativos:

```
10 PRINT"VALOR DE X","NUMERO"
20 PRINT
30 FOR X=-1 TO -20 STEP -1
40 PRINT X,RND(X)
50 NEXT
```

Si X es cero, la función RND toma como valor el anterior número obtenido al azar. Puedes comprobarlo con:

```
10 INPUT "VALOR DE X";X
20 PRINT RND(X),RND(0)
30 GOTO 10
```

La instrucción RND, como hemos visto hasta ahora, obtiene números aleatorios de una lista prefijada de antemano. Precisamente

por esto, es relativamente fácil conocer los números que, en cada caso, se obtengan con esta instrucción. Debido a esto, se pierde el carácter de aleatoriedad que debe exigirse en muchos programas.

Hay una instrucción que soluciona con bastante eficacia esta cuestión:

RANDOMIZE

que admite dos modalidades:

- RANDOMIZE X

Que toma los valores aleatorios de la lista, a partir del que ocupa el lugar número X.

- RANDOMIZE TIME

Que toma los valores a partir de una posición determinada por el tiempo que lleva conectado el aparato.

SIMULACIONES

Uno de los usos más habituales de la función RND es la simulación de diferentes experimentos. Así, el experimento "lanzar una moneda" podemos considerarlo equivalente a obtener un número aleatorio comprendido entre 0 y 1. Si el número es menor que 0.5 diremos que ha salido cara; en caso contrario, se considerará cruz.

Un programa que presente en pantalla los resultados obtenidos al lanzar 20 veces una moneda, será:

```
10 RANDOMIZE TIME
20 FOR X=1 TO 20
30 M=RND
40 IF M<0.5 THEN PRINT"CARA" ELSE PRINT"
CRUZ"
50 NEXT
```

Puedes modificar el programa de forma que pida el número de lanzamientos y contabilice el número de caras y cruces aparecidos.

En muchas simulaciones el número que debe obtenerse al azar debe ser entero y estar comprendido entre dos números P y Q. Podemos conseguirlo con:

INT ((Q−P+1) * RND)+P

Por ejemplo, si queremos simular el lanzamiento de un dado, la orden sería:

INT (6 * RND)+1

Si deseamos simular el juego de la ruleta, tendremos que obtener un número entero comprendido entre 0 y 36, ambos inclusives; por consiguiente, la orden será:

INT (37 * RND)

Con el siguiente programa podrás obtener 10 números enteros al azar comprendidos entre dos valores cualesquiera P y Q.

```
10 INPUT "DAME EL NUMERO MENOR";P
20 INPUT "DAME EL NUMERO MAYOR";Q
30 RANDOMIZE TIME
40 FOR X=1 TO 10
50 PRINT INT((Q-P+1)*RND)+P
60 NEXT
```

Ahora, utilizando las instrucciones anteriores, estás en disposición de realizar programas que simulen los siguientes experimentos:

- Lanzar 1000 veces un dado y contabilizar las apariciones de cada una de las puntuaciones.
- Obtener quinielas futbolísticas al azar. Puedes dar diferentes probabilidades al 1, X y 2, con el fin de acercarlas más a la realidad.
- Simular la extracción de los tres primeros premios de la lotería. Ten presente que deben ser números de 5 cifras y el 0 puede ser cualquiera de ellas.
- Rellenar al azar una apuesta de la Lotería Primitiva. Deberás presentar correctamente la apuesta en pantalla, teniendo presente que los 6 números han de ser distintos.
- Lanzar una moneda repetidamente hasta que se obtenga un número, prefijado de antemano, de caras consecutivas.

En ese momento se indicará el número de caras y cruces que han aparecido.

- Un jugador de baloncesto tiene una probabilidad del 78% de acertar tiros libres. Simular el lanzamiento de 100 tiros libres.
- Calcular por simulación la probabilidad de que dos cazadores disparando un tiro cada uno, maten un conejo si el primero acierta un disparo de cada tres y el segundo uno de cada cinco.
- Comprobar por simulación el siguiente resultado:

Tenemos una baraja española de 40 cartas y disponemos de 100 pesetas. Extraemos una carta al azar. Si sale oros o copas, ganamos la mitad de lo que tenemos y si sale espadas o bastos perdemos la mitad de nuestro dinero. Una vez extraídas todas las cartas, acabamos siempre con la misma cantidad de dinero en nuestro poder, independientemente del orden de extracción de las cartas.

JUEGOS

En este apartado presentaremos dos juegos sencillos basados en la instrucción RND y a continuación te propondremos unos cuantos que sirvan de base para imaginar y construir tus propios programas sobre este tema.

Juego 1

El ordenador tomará al azar un número entero comprendido entre 1 y 100. Debes encontrar dicho número en un máximo de seis intentos. En cada uno de éstos, el ordenador informará si el número por él elegido es mayor menor o igual que el que tú has introducido.

```
10 RANDOMIZE TIME
20 X=INT(100*RND)+1
30 FOR Y=1 TO 6
40 INPUT "DAME UN NUMERO";N
50 IF X>N THEN PRINT N;"ES MENOR"
60 IF X<N THEN PRINT N;"ES MAYOR"
70 IF X=N THEN PRINT"ACERTO EN";Y;"INTEN
TOS":END
80 NEXT
90 PRINT"EL NUMERO ERA";X
```

Juego 2

Este juego es muy conocido y recibe el nombre de MASTER–MIND. Consiste en que el ordenador elige al azar un número de cuatro cifras todas ellas distintas y la primera no nula. Nosotros debemos adivinar dicho número en un máximo de nueve intentos. Tras cada uno de éstos el ordenador nos indicará la cantidad de cifras que hemos acertado en la misma posición (muertos) y aquellas que han sido acertadas pero se han colocado en distinta posición (heridos).

```
10 RANDOMIZE TIME
20 MODE 1
30 A=INT(9*RND)+1
40 B=INT(10*RND)
50 C=INT(10*RND)
60 D=INT(10*RND)
70 IF A=B OR A=C OR A=D OR B=C OR B=D OR
 C=D THEN 30
80 FOR X=1 TO 9
90 INPUT "ESCRIBE LAS CUATRO CIFRAS";U,V
,X,Y
100 M=0:H=0
110 IF A=U THEN M=M+1
120 IF B=V THEN M=M+1
130 IF C=X THEN M=M+1
140 IF D=Y THEN M=M+1
150 IF A=V OR A=X OR A=Y THEN H=H+1
160 IF B=U OR B=X OR B=Y THEN H=H+1
170 IF C=U OR C=V OR C=Y THEN H=H+1
180 IF D=U OR D=V OR D=X THEN H=H+1
190 PRINT M;"MUERTOS",H;"HERIDOS"
200 IF A=U AND B=V AND C=X AND D=Y THEN
PRINT"ACERTO EN";N;"INTENTOS":END
210 NEXT
220 PRINT"EL NUMERO ERA";A;B;C;D
```

Ahora puedes intentar hacer los siguientes:

- Un tesoro se encuentra escondido en una de las casillas en que se encuentra dividida la pantalla (El número de casillas dependerá del modo de trabajo). Nuestra misión será encontrarlo en 15 intentos dando la fila y la columna correspondiente a la casilla del tesoro. Tras cada intento el ordenador especificará si nos encontramos muy cerca, cerca, lejos o muy lejos de él. (Establece el criterio de proximidad a tu gusto).

- Este es una variante del anterior. En lugar del tesoro supongamos que tenemos una rana juguetona que, tras cada intento de adivinación de su posición, si no es capturada, salta a una casilla contigua al azar. En lugar de indicar un mensaje de proximidad se puede dar la distancia entre la casilla donde está situada la rana y la casilla designada por nosotros.
- El siguiente juego es el inverso del JUEGO 1. En este caso nosotros elegimos el número comprendido entre 1 y 100 y el ordenador deberá encontrarlo. El método que utilizará el ordenador se basará en aproximaciones al azar. Para indicar al ordenador que su número es mayor, menor o igual que el elegido por nosotros, pulsaremos las teclas 1, 2 ó 3 respectivamente.
- El ordenador presenta un número de 6 cifras en pantalla durante un cierto tiempo. El número, su posición y el tiempo de presentación se tomarán al azar. Una vez borrado este número, debemos escribirlo nosotros y el ordenador nos dará un mensaje de correcto o incorrecto. Podemos hacer que el juego se repita un número de veces y contabilizar el número de aciertos. (El tiempo de permanencia en pantalla del número en cuestión se conseguirá con un bucle vacío).
- Un juego de dados muy conocido es el llamado "la jaula". Consiste en lanzar tres dados apostando previamente por un número del 1 al 6. Si el número elegido coincide con el resultado obtenido en un solo dado, ni ganamos, ni perdemos. Si coincide en dos dados, ganamos lo apostado y si tenemos la suerte de que coincida con los tres, ganamos el doble de lo apostado. Lógicamente si nuestro número no coincide con ningún dado, perdemos la apuesta.
  Inicialmente disponemos de 1000 ptas. y el ordenador debe presentar el desarrollo del juego y contabilizar nuestras ganancias y pérdidas.

10
Sonido

Una de las características más brillantes de nuestro AMSTRAD es su calidad musical. Con él conseguiremos componer todo tipo de música de una manera bastante convincente.

Todas las órdenes referidas al sonido se procesan en el circuito integrado AY – 3 – 8912 que podemos considerar como independiente dentro del ordenador.

Las doce notas que maneja nuestro ordenador son:

DO, DO#/REb, RE, RE#/MIb, MI, FA, FA#/SOLb, SOL, SOL#/LAb, LA, LA#/SIb, SI, DO

donde el símbolo # indica sostenido y b bemol.

La orden fundamental para trabajar el sonido es:

SOUND

Esta orden puede admitir hasta siete parámetros

SOUND CA, T, D, V, EV, ET, R

cuyo significado es:

PARAMETRO	SIGNIFICADO
CA	Canal
T	Tono
D	Duración
V	Volumen
EV	Envolvente de volumen
ET	Envolvente de tono
R	Ruido

Solamente los dos primeros son imprescindibles.

En este apartado analizaremos el primero de ellos aunque se utilizarán alguno de los otros parámetros.

El AMSTRAD posee tres canales distintos de emisión de sonido que llamaremos A, B y C. El parámetro CA indicará por qué canal o canales se emitirá la nota. Además, con este parámetro, podemos realizar dos interesantes operaciones que posteriormente comentaremos: retención de sonido y borrado de cola.

La siguiente tabla presenta los efectos que podemos conseguir según los valores del parámetro CA.

VALOR DE CA	EFECTO
1	Elegir canal A
2	Elegir canal B
4	Elegir canal C
8	Sincronizar con A
16	Sincronizar con B
32	Sincronizar con C
64	Retener sonido
128	Borrar cola de sonido

Si queremos conseguir varios de estos efectos a la vez, bastará con dar a CA el valor que resulta de sumar los valores correspondientes a dichos efectos. Así, si

CA = 76

estamos seleccionando el canal C (valor 4), sincronizándolo con el canal A (valor 8) y reteniendo el sonido (valor 64).

Analicemos ahora las funciones de retención y borrado de cola que resultan menos familiares para el lector.

– Retención de sonido

Si en el valor de CA hemos incluido esta opción, la nota correspondiente no sonará hasta que se libere con la orden

RELEASE K

donde K es un número comprendido entre 1 y 7 y que se rige de acuerdo con la siguiente tabla:

VALOR DE K	EFECTO
1	Libera el canal A
2	Libera el canal B
3	Libera los canales A y B
4	Libera el canal C
5	Libera los canales A y C
6	Libera los canales B y C
7	Libera los canales A, B y C

Puedes observar el efecto de esta orden ejecutando el siguiente programa

```
10 SOUND 66,478,500,7
20 SOUND 1,253,500,7
```

y añadiendo a continuación la línea

```
12 RELEASE 2
```

– Borrado de cola

Como ya se ha dicho anteriormente, el sonido se trabaja en un circuito casi independiente. Esto implica que las diversas órdenes de sonido pueden procesarse y quedar a la espera de ser emitidas. Esta espera ocasiona lo que denominamos cola, que puede constar de un máximo de cinco sonidos. Estos sonidos se emitirán uno tras otro, según el orden en que están en la cola.

Si en algún caso nos interesase eliminar estos sonidos almacenados, lo que llamamos borrado de cola, deberemos añadir al valor de CA el número 128, como se ha indicado anteriomente.

Puedes comprobar como funciona el borrado de cola con el siguiente programa:

```
10 SOUND 2,63,500,7
20 SOUND 1,95,500,7
30 SOUND 1,213,500,7
40 SOUND 1,239,500,7
50 SOUND 1,190,500,7
```

En los cinco primeros segundos (valor 500) se emiten simultáneamente dos notas; una por el canal B (línea 10) y otra por el canal A (línea 20). A continuación, se emitirán, con una duración de cinco segundos cada una, tres notas más por el canal A (líneas 30, 40, y 50). Estas tres últimas notas estaban aguardando en la cola de espera del canal A. Por tanto, si sustituimos la línea 40 por

```
40 SOUND 129,239,500,7
```

se borran los sonidos correspondientes a las líneas 20 y 30.

Para saber si existe cola en un canal, hay que utilizar la instrucción

SQ (K)

donde K toma los valores 1, 2 ó 4 según que nos refiramos al canal A, B ó C respectivamente.

Cuando SQ (K) sea mayor que 127, el canal indicado por K poseerá cola. Esta propiedad será de gran utilidad cuando queramos detener un programa mientras queden sonidos en la cola.

NOTAS MUSICALES

El segundo parámetro de la instrucción SOUND indica la nota que queremos emitir. El valor de T vendrá dado por un número comprendido entre 0 y 4095.

Puedes comprobar la influencia de este parámetro con:

```
10 INPUT "VALOR DE T";T
20 SOUND 1,T,500,7
30 WHILE SQ(1)>127:WEND
40 GOTO 10
```

Observa que la línea 30 se ha colocado con el fin de que pida un nuevo valor de T una vez acabado de emitirse el sonido.

Los valores que hay que dar a T para obtener las notas deseadas vienen dados por la expresión.

T = ROUND (125000/F)

donde

F = 440 * (2 ↑ (O + (S − 10) / 12))

siendo O el número de octava deseada (un valor entero comprendido entre −3 y 4, ambos inclusive) y S el número de semitono (dentro de dicha octava) de acuerdo con la siguiente tabla:

NOTA	SEMITONO
DO	1
DO # /REb	2
RE	3
RE # /MIb	4
MI	5
FA	6

FA#/SOLb	7
SOL	8
SOL#/LAb	9
LA	10
LA#/SIb	11
SI	12

Ahora ya estamos en condiciones de construir una escala musical.

```
10 INPUT "NUMERO DE OCTAVA";O
20 FOR N=1 TO 8
30 READ S
40 F=440*(2^(O+(S-10)/12))
50 T=ROUND (125000/F)
60 SOUND 1,T
70 NEXT
80 DATA 1,3,5,6,8,10,12,13
```

El tercer parámetro (D) indica la duración de la nota en centésimas de segundo. Asi,

D = 1000

obligará a que la nota se emita durante un tiempo de 10 segundos. En caso de no consignarse el valor de D, automáticamente se considerará

D = 20

El valor de D debe estar comprendido entre −32768 y 32767. Si D = 0 la duración de la nota está controlada por la envolvente de volumen (apartado siguiente) mientras que si D es negativo, indica el número de veces que se repite de acuerdo con la antedicha envolvente de volumen.

El cuarto parámetro (V) es el encargado de controlar el volumen de emisión. Su valor está comprendido entre 0 (volumen mínimo) y 7 (volumen máximo) cuando D es positivo. Si D es cero, el volumen también está controlado por una envolvente de volumen y el valor de V se refiere a la envolvente que se está considerando. En este caso V podrá tomar valores entre 0 y 15.

Por último, antes de pasar al estudio de las envolventes, analizaremos el parámetro final (R) de la instrucción SOUND. Este parámetro, que debe estar comprendido entre 0 y 31, se utiliza especialmente cuando queremos conseguir determinados efectos especiales. Compruébalo con

```
10 INPUT "VALOR DE R";R
20 SOUND 1,478,500,7,0,0,R
30 WHILE SQ(1)>127:WEND
40 GOTO 10
```

Puedes observar que si asignamos a R el valor 0, la nota suena limpia.

ENVOLVENTES DE SONIDO

La envolvente de volumen controla los diferentes volúmenes que puede tener una nota mientras está sonando. El periodo de emisión de la nota puede descomponerse hasta en cinco secciones, cada una de las cuales viene determinada por tres valores: el número de escalones, la altura y duración de cada uno de ellos.

La envolvente de volumen se define con la orden

ENV

que, según lo dicho anteriormente, puede tener hasta 16 parámetros: uno para indicar el número de envolvente y los 15 restantes para definir las cinco secciones. Por tanto, su forma general será:

$$\text{ENV } N, A_1, B_1, C_1, A_2, B_2, C_2, A_3, B_3, C_3, A_4, B_4, C_4, A_5, B_5, C_5$$

donde N es el número de la envolvente y A_i, B_i, C_i (i = 1,2,3,4,5) son respectivamente el número de escalones, su altura y su duración.

Observa como funciona la envolvente de volumen con el siguiente programa, en el cual solamente hemos definido una sección:

```
10 ENV 1,6,2,50
20 SOUND 1,478,0,0,1
```

El quinto parámetro de SOUND es 1 para indicar que el volumen y la duración de la nota están controlados por la envolvente número 1 que está definida en la línea 10.

Los parámetros A_i, B_i, C_i deben estar comprendidos entre:

PARAMETRO	MINIMO	MAXIMO
A_i	0	127
B_i	−128	127
C_i	0	255

El parámetro que nos falta por analizar de la instrucción SOUND, es ET, que recibe el nombre de envolvente de tono. Su funcionamiento es análogo al de la envolvente de volumen con la diferencia de que el control se ejerce ahora sobre el tono.

La definición de la envolvente de tono se efectúa con la orden

ENT

que, como en el caso de ENV, también puede tener hasta 16 parámetros: uno para indicar el número de envolvente y los otros quince para designar hasta cinco secciones.

ENT N, D_1, F_1, G_1, D_2, F_2, G_2, D_3, F_3, G_3, D_4, F_4, G_4, D_5, F_5, G_5

Estos parámetros deben estar comprendidos según la tabla

PARAMETRO	MINIMO	MAXIMO
D_i	0	127
F_i	−128	127
G_i	0	239

11
Gráficos

En este capítulo estudiaremos las posibilidades gráficas del ordenador AMSTRAD pero excluyendo todo lo referente al color, que ya se verá más adelante.

Cuando dibujamos sobre la pantalla del ordenador decimos que estamos empleando la "pantalla gráfica" a diferencia de la "pantalla de texto" usada hasta ahora.

El sistema de referencia de dicha "pantalla gráfica" esta situado en la parte inferior izquierda de PAPER de tal manera que consideramos que un punto tiene de coordenadas x e y si nos hemos desplazado x unidades horizontalmente e y unidades verticalmente respecto del origen.

Para dibujar un punto en la pantalla se usa la instrucción:

- PLOT

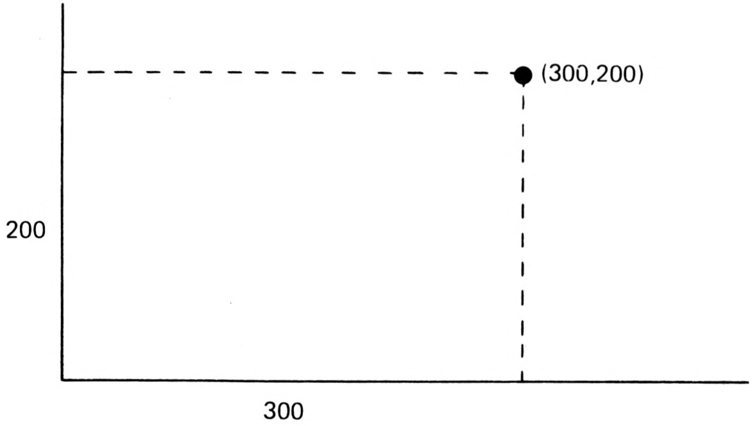
Que tiene la forma general:

```
PLOT x,y
```

donde x e y son las coordenadas del punto. Así:

```
PLOT 300, 200
```

dibujará el punto (300, 200).



Los puntos situados dentro de la pantalla son aquellos que tienen la primera coordenada comprendida entre 0 y 639 y la segunda entre 0 y 399, ambos inclusive. Por lo tanto, con el programa:

```
10 MODE 0
20 PLOT 0,0:PLOT 639,0
30 PLOT 0,399:PLOT 639,399
```

se dibujarán cuatro puntos situados en los cuatro vértices de PAPER.

La instrucción PLOT admite coordenadas fuera de los límites señalados anteriormente, aunque no tenga efectos visibles en la pantalla.

Es posible que el lector haya sacado la conclusión, por lo expuesto anteriormente, de que es factible dibujar:

640x400=256000

puntos dentro de la zona de PAPER. Pues bien, esto no es cierto por lo que explicaremos ahora.

Vamos a considerar cuadriculada la pantalla dividiendo el eje x en 640 partes iguales, numeradas del 0 al 639, y el eje y en 400, numeradas del 0 al 399. En total: 640 x 400 = 256000 cuadrículas. Sin embargo, nosotros no podremos dibujar en cada una de estas cuadrículas un punto, por las dos razones que damos a continuación:

1– Cada vez que se dibuja un punto se ocupan dos cuadrículas, una encima de la otra, de las mencionadas anteriormente. Así, con:

PLOT 0,0 y PLOT 1,0

no se dibujan dos puntos diferentes, ya que en cada caso se ocupan las mismas cuadrículas. Comprobarlo con el programa:

```
10 MODE 1
20 FOR Y=0 TO 399
30 PLOT 0,Y
40 A$=INKEY$:IF A$="" THEN 40
50 NEXT
```

en el que son precisas dos pulsaciones para dibujar un único punto en la pantalla.

Una línea vertical visible en la pantalla estará formada, como máximo, por 200 puntos, aunque su coordenada "y" vaya de 0 a 399. Además hay que añadir que esto ocurre cualquiera que sea el MODE utilizado.

2– El número de puntos que se pueden dibujar sobre la horizontal depende del MODE elegido. Así:

– En MODE 2 existe la posibilidad de dibujar 640 puntos que corresponde a la máxima resolución. Cada punto ocupará una cuadrícula horizontalmente.

– En MODE 1 son 320 los puntos que podremos dibujar, ya que, por ejemplo, con:

PLOT 0,0 y PLOT 0,1

dibujamos el mismo punto. En este caso, cada punto ocupa dos cuadrículas horizontalmente.

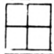
– En MODE 0 son solamente 160, ya cada punto ocupa cuatro cuadrículas horizontalmente. Así se comprende que:

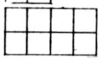
PLOT 0,0 , PLOT 0,1 , PLOT 0,2 y PLOT 0,3

dibujen el mismo punto.

De forma gráfica se puede poner el tamaño del punto según el MODE en que se trabaje:

En MODE 2 

En MODE 1 

En MODE 0 

donde cada cuadrado elemental corresponde a una cuadrícula de las mencionadas anteriormente.

Comprobar lo dicho con el siguiente programa:

```
10 INPUT "MODE";M
20 MODE M
30 FOR X=0 TO 639
40 PLOT X,0
50 A$=INKEY$:IF A$="" THEN 50
60 NEXT
```

y observar que se necesitan 1, 2 o 4 pulsaciones para cada punto dibujado según se trabaje en MODE 2, MODE 1 o MODE 0 respectivamente.

En resumen, en cada modo es posible representar en la pantalla el número de puntos que se indica a continuación:

En MODE 0 160x200=32000
En MODE 1 320x200=64000
En MODE 2 640x200=128000

–PLOTR

Es una instrucción muy parecida a la anterior pero trabaja con coordenadas relativas. Esto quiere decir que si el último punto dibujado ha sido en las coordenadas x e y, con:

PLOTR a,b

se dibujará otro punto desplazado "a" unidades horizontalmente y "b" unidades verticalmente respecto del último punto dibujado. Si "a" es negativo,el desplazamiento se hace a la izquierda y si "b" es negativo,hacia abajo.

Por ejemplo, con:

PLOT 100, 100

hemos dibujado un punto de coordenadas (100, 100). Con

PLOTR 40, –50

habremos dibujado otro punto desplazado, respecto del anterior, 40 unidades hacia la derecha y 50 unidades hacia abajo. El mismo efecto hubieramos obtenido con:

PLOT 140, 50

También es posible dibujar rectas con la instrucción:

– DRAW

Cuya forma general es:

DRAW x,y

y tiene el efecto de unir el último punto dibujado, en su defecto el (0,0), con el punto de coordenadas x,y.

Por ejemplo, se consigue dibujar el contorno de PAPER con:

```
10 MODE 2
20 DRAW 639,0:DRAW 639,399
30 DRAW 0,399:DRAW 0,0
```

Con esta instrucción se puede unir puntos que no aparezcan en la pantalla sin que el ordenadro dé mensaje de error. Así, puedo unir el punto de coordenadas (0,0) con el punto de coordenadas (1000,1000) de la siguiente manera:

```
10 PLOT 0,0
20 DRAW 1000,1000
```

Una instrucción parecida a la anterior es:

– DRAWR

Donde, como cabe esperar, se trabaja con coordenadas relativas. Con:

DRAWR a,b

se une con una recta el último punto dibujado y otro punto desplazado "a" unidades horizontalmente y "b" unidades verticalmente. Si "a" es negativa, el desplazamiento horizontal es hacia la izquierda y si "b" es negativa, el desplazamiento es hacia abajo. Así, con:

```
10 PLOT 120,150
20 DRAWR -20,40
```

se unen entre sí, con una recta, los puntos de coordenadas (120, 150) y (100, 190).

Veamos como se dibuja el contorno de PAPER con DRAWR:

```
10 MODE 2
20 DRAWR 639,0:DRAWR 0,399
30 DRAWR -639,0:DRAWR 0,-399
```

De forma análoga al "CURSOR DE TEXTO" visto con anterioridad, también se puede hablar de "CURSOR GRAFICO" que, aunque no sea visible, siempre estará situado en el último punto dibujado. Para mover este cursor se emplea la instrucción:

– MOVE

Su forma general es:

MOVE x,y

y coloca el cursor gráfico en el punto de coordenadas x e y. Puede servir para colocar el cursor gráfico en un punto sin necesidad de dibujar ningún punto.

– MOVER

Su forma general es:

MOVER x,y

tiene el mismo efecto que la anterior instrucción pero empleando coordenadas relativas. Con:

MOVER x,y

se logra desplazar el cursor gráfico "x" unidades en sentido horizontal e "y" unidades en sentido vertical.

Para conocer las coordenadas del cursor gráfico se usa las instrucciones:

- XPOS, YPOS

Con:

```
PRINT XPOS; YPOS
```

se imprimen en la pantalla las dos coordenadas de la posición del cursor gráfico. Esta instrucción puede llegar a ser útil cuando no sabemos donde se encuentra dicho cursor.

- ORIGIN

Inicialmente el origen de coordenadas en la pantalla gráfica ya hemos dicho que se encuentra en la parte inferior izquierda de PAPER. Pero también esto se puede modificar con la instrucción ORIGIN. Así:

```
ORIGIN x,y
```

sitúa el nuevo origen en el punto de coordenadas (x,y). Con:

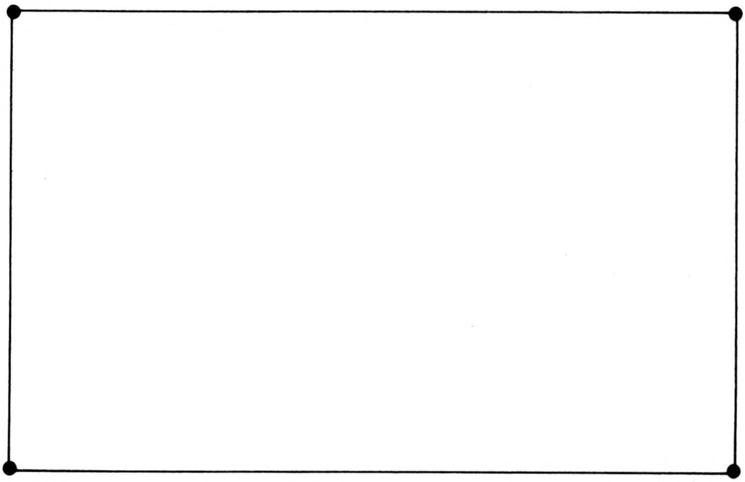
```
ORIGIN 319,199
```

situamos el origen en el centro, aproximadamente, de la pantalla, por lo que los 4 vértices de PAPER tendrán ahora las coordenadas que aparecen en la página siguiente.

Al usar la instrucción MODE se vuelve a situar el origen en su posición inicial.

- CLG

Esta instrucción borra la pantalla gráfica, por lo que se parece mucho a la instrucción CLS. También coloca el origen gráfico en su po-



sición inicial. Al tratar el color, se verá otra manera de usar esta instrucción que resultará muy interesante.

– TAG–TAGOFF

En algunas ocasiones, sobre todo cuando hay que mezclar gráficos y texto en la pantalla, será interesante escribir caracteres en la posición del cursor gráfico en lugar de la posición del cursor de texto, empleando la instrucción TAG.

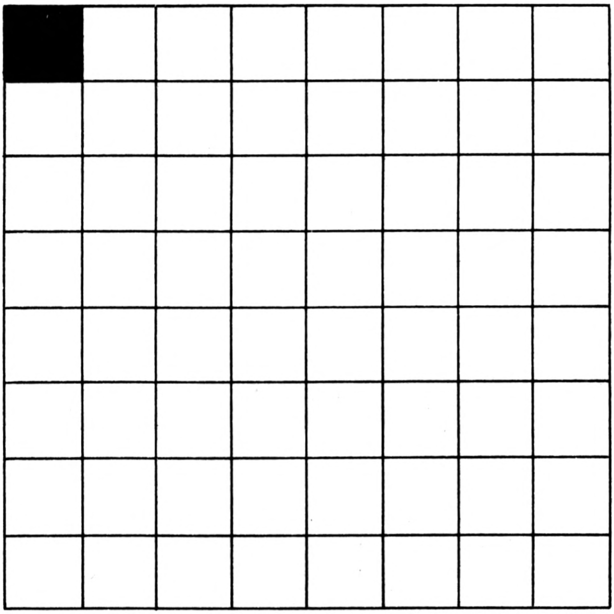
Con el programa:

```
10 MODE 1
20 LOCATE 20,12
30 ?"XY"
```

hemos escrito los caracteres "XY" centrados en la pantalla. Hasta ahora no nos era posible escribir en una posición intermedia entre las posiciones ocupadas por ambos caracteres, pero con esta nueva instrucción esto si será posible. Primero coloco el cursor gráfico en la posición deseada y, después de TAG, ya puedo escribir en esa posición del cursor gráfico.

```
40 MOVE 310,224
50 TAG
60 PRINT "A";
```

con el cursor gráfico se señala la parte superior izquierda del espacio que va a ocupar el nuevo carácter a imprimir.



En nuestro ejemplo, se consigue escribir entre los dos caracteres "XY" el carácter "A" que, por supuesto borrará parte de los dos anteriores.

Es necesario colocar el punto y coma de la línea 60, ya que, en caso contrario, aparecerán signos indeseados.

Con TAGOFF se logra anular el efecto de TAG, por lo que todo lo que se escriba a partir de este momento será en la posición del cursor de texto.

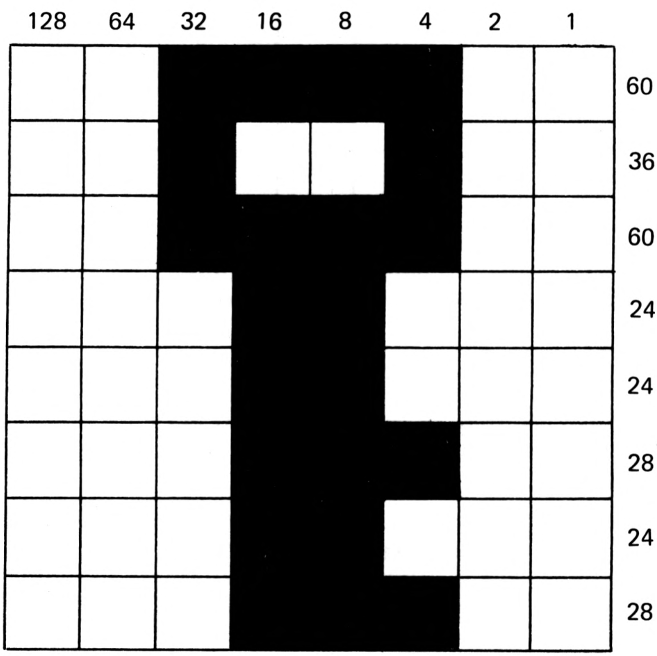
GRAFICOS DEFINIDOS PARA EL USUARIO

Ya hemos visto que cada carácter que se puede presentar en la pantalla del ordenador lleva asociado un número, llamado código ASCII, mediante el cual se le identifica. Por ejemplo, el carácter " " tiene asociado el código 240, por lo que con:

```
PRINT CHR$(240)
```

dicho carácter se imprime en la pantalla.

Pero también es posible asignar ese código ASCII a un nuevo carácter que nosotros hayamos diseñado. Para ello, disponemos de una cuadrícula 8x8 sobre la cual se puede ennegrecer o dejar en blanco cada uno de los 64 cuadraditos que la componen. Por ejemplo, se puede diseñar el gráfico de la figura:



Para que el ordenador memorice ese carácter, se le tiene que suministrar un número por cada fila. Ahora veremos como se obtiene.

Para cada fila se suman los números que aparecen en la parte superior de los diferentes cuadraditos ennegrecidos. En nuestro ejemplo:

```
FILA 1:   32+16+8+4=60
FILA 2:   32+4=36
FILA 3:   32+16+8+4=60
FILA 4:   16+8=24
FILA 5:   16+8=24
FILA 6:   16+8+4=28
FILA 7:   16+8=24
FILA 8:   16+8+4=28
```

estos 8 números se le suministrarán al ordenador con la instrucción:

– SYMBOL

De la siguiente manera:

```
SYMBOL 240,60,36,60,24,24,28,24,28
```

el primer número es el código ASCII asignado al nuevo carácter y los 8 restantes son los calculados anteriormente.

Para imprimir en la pantalla el nuevo carácter, se emplea la instrucción:

```
PRINT CHR$(240)
```

– SYMBOL AFTER

Por el procedimiento anterior sólo se puede redefinir los caracteres cuyos códigos van del 240 hasta el 255. Pero si empleamos la instrucción:

```
SYMBOL AFTER b
```

se podrán redefinir los caracteres asignados a los códigos entre b y 255, siendo b, como mínimo, 32.

El máximo número de caracteres redefinibles se obtienen con:

```
SYMBOL AFTER 32
```

12
Color

Con el ordenador AMSTRAD, y siempre que se conecte un monitor en color, se dispone de una paleta de 27 colores aunque, como veremos más adelante, no es posible presentarlos todos ellos simultáneamente en la pantalla.

A cada color se le asocia un número, entre 0 y 26, que llamaremos CODIGO DE COLOR, según la tabla siguiente:

CODIGO	COLOR	CODIGO	COLOR
0	NEGRO	14	AZUL PASTEL
1	AZUL	15	NARANJA
2	AZUL BRILLANTE	16	ROSA
3	ROJO	17	MAGENTA PASTEL
4	MAGENTA	18	VERDE BRILLANTE
5	MALVA	19	VERDE MARINO
6	ROJO BRILLANTE	20	CYAN BRILLANTE
7	PURPURA	21	VERDE LIMA
8	MAGENTA BRILLANTE	22	VERDE PASTEL
9	VERDE	23	CYAN PASTEL
10	CYAN	24	AMARILLO BRILLANTE
11	AZUL CIELO	25	AMARILLO PASTEL
12	AMARILLO	26	BLANÇO BRILLANTE
13	BLANCO		

Inicialmente, las dos zonas en que está dividida la pantalla, BORDER y PAPER, aparecerán coloreadas de azul (código 1) mientras que la tinta con la que se escribe el texto es amarilla (código 24).

A partir de este momento veremos cómo se manejan las instrucciones relacionadas con el color. Empezaremos por la más sencilla:

– BORDER

De forma general:

BORDER n

donde n es el código de color elegido para colorear la zona BORDER de la pantalla:

Con:

BORDER 0

colorearemos dicha zona de color negro,ya que el 0 es el código asociado a ese color. El resto de la pantalla no sufre ninguna alteración.

Otra manera de usar esta instrucción es la siguiente:

BORDER m,n

Donde m y n son los códigos de dos colores. El efecto de esta instrucción es colorear el borde de la pantalla con los colores m y n alternativamente.

Así:

BORDER 15,22

colorea el borde de la pantalla de NARANJA (código 15) y VERDE PASTEL (código 22) de forma alternativa.

El tiempo que permanece cada color, se puede controlar con la instrucción:

– SPEED INK

De forma general:

```
SPEED INK a,b
```

en donde a y b marcan el tiempo de permanencia de cada color medidos en unidades de 0.02 segundos.

Por ejemplo, con las instrucciones:

```
SPEED INK 10,20
BORDER 15,22
```

se consigue que el borde de la pantalla se coloree oscilando entre el color NARANJA y VERDE PASTEL, manteniéndose el primero 0.2 seg. y el segundo 0.4 seg.

A partir de este momento nos ocuparemos del resto de la pantalla para poder elegir, dentro de nuestras posibilidades, el color de PAPER y el de la tinta con la que escribimos.

Las cosas suceden como si dispusiésemos de:

16 tinteros en MODE 0

4 tinteros en MODE 1

2 tinteros en MODE 2.

y cada uno de los tinteros podrá estar lleno de cualquiera de las 27 tintas ya mencionadas (códigos 0-26).

Al conectar el ordenador, la distribución de las tintas en los diferentes tinteros es como la que se muestra en la página siguiente.

Como se verá en el apartado siguiente, esta distribución de tintas se puede cambiar fácilmente llenando cada tintero con la tinta deseada.

En realidad, y en cualquier MODE en que trabajemos, es 16 el número de tinteros disponibles (numerados del 0 al 15), pero:

En MODE 2 la tinta de todos los tinteros pares es siempre la misma que la del tintero 0 y todos los tinteros impares contienen siempre el mismo color de tinta. Por lo que en la práctica sólo se dispone de dos tinteros en este MODE.

TINTERO	CODIGOS DE COLORES		
	MODE 0	MODE 1	MODE 2
0	1	1	1
1	24	24	24
2	20	20	
3	6	6	
4	26		
5	0		
6	2		
7	8		
8	10		
9	12		
10	14		
11	16		
12	18		
13	22		
14	1,24		
15	16,11		

En MODE 1, los grupos de tinteros:

4, 5, 6 y 7
8, 9, 10 y 11
12, 13, 14 y 15

contiene siempre la misma tinta que los tinteros:

0, 1, 2 y 3

respectivamente. Es decir, el color de tinta se repite cada 4 tinteros, por lo que son cuatro los tinteros disponibles.

En la tabla anterior, y en MODE 0, se observa que los tinteros números 14 y 15 están llenos con dos tintas, lo cual quiere decir que el color resultante oscila entre ambos colores y a una velocidad que se controla con la instrucción SPEED INK.

Inicialmente, en cualquier MODE que trabajemos, los tinteros 0 y 1 contienen las tintas de códigos 1 y 24 respectivamente. En el tintero número 0 se guarda la tinta con la que se colorea el fondo de PAPER y en el tintero número 1 se guarda la tinta con la que escribimos. Por lo tanto, y tal como comentábamos anteriormente, el fondo es de color AZUL y la tinta con la que se escribe es AMARILLA.

– INK

De forma general:

INK a,b

tiene como efecto el llenar el tintero número a con la tinta de código b. Por ejemplo, con:

INK 0,4   INK 1,15

llenaremos los tinteros 0 y 1 con los colores de tinta 4 y 15 respectivamente; con ello conseguiremos que el color del fondo de PAPER pase a MAGENTA (código 4) y el de tinta para escritura a NARANJA (código 15), en cualquier MODE en que trabajemos. Además se debe observar que el cambio de tinta en el tintero 1 afecta a todo el color del texto escrito hasta ese momento.

No siempre el cambio de tinta de un tintero tiene efectos visibles en la pantalla. Así, con el programa:

```
10 MODE 1
20 INK 3,5
30 INK 4,6
```

se cambia la tinta de los tinteros 3 y 4 sin que esto tenga ningún efecto visible, quedando estas tintas a nuestra disposición para una posterior utilización.

Otra versión de la instrucción INK es:

INK a,b,c

con la que llenamos el tintero a con las tintas b y c, de tal manera que el color de la tinta resultante oscila entre ambos colores.

Con:

INK 0,2,10

conseguimos que la tinta con la que coloreamos el fondo de la pantalla, parpadee entre los colores: AZUL BRILLANTE (código 2) y CYAN (código 10). La velocidad de este parpadeo, como en casos anteriores, se controla con la instrucción SPEED INK m,n.

– PEN

Empleando:

PEN n

conseguiremos mojar la pluma, con la que escribimos en la pantalla, en el tintero n que previamente ha podido ser llenado con el color de tinta deseado o dejarlo con el color que se le asigna inicialmente.

El parámetro n puede tomar valores entre 0 y 15, pero recordemos que:

- En MODE 2 todos los tinteros pares contienen el mismo color de tinta.
- En MODE 1 cada 4 tinteros se repite el color de la tinta que contienen.

Teniendo en cuenta que si la escritura tiene que ser legible, el color de la tinta tendrá que ser diferente del color del fondo; así, en:

MODE 0	podremos escribir con 15 colores diferentes de tinta.
MODE 1	podremos escribir con 3 colores diferentes de tinta.
MODE 2	sólo con 1.

Al cambiar la tinta de un tintero, todo lo escrito con la tinta de ese tintero hasta ese momento, cambia al nuevo color.

Esta instrucción se puede emplear de otra manera:

PEN a,b

donde a es el número de un tintero y b puede tomar los valores 0 y 1.

Si b=1, se puede escribir en una posición de la pantalla sin borrar lo escrito en esa posición con anterioridad (forma transparente). Si b=0 se escribe normal.

- PAPER

De forma general:

PAPER n

donde n es el número de un tintero. Con ello coloreamos el fondo de los caracteres que se imprimen en la pantalla. Por eso, en MODE 1 o en MODE 0, con:

PAPER 3

se colorea, si el tintero número 3 contiene su tinta primitiva, el fondo de cada carácter de color ROJO.

Para conseguir que este color se extienda por todo PAPER es preciso usar la instrucción CLS.

13
Grabación y carga de programas

A estas alturas resultará evidente para el lector la necesidad que existe de un dispositivo que nos sirva para almacenar, de forma permanente, programas y datos interesantes de guardar.

Tendremos que acudir a memorias externas al ordenador para que la información no se pierda cuando se desconecte de la red. Podrá hacerse sobre una simple cinta casete (en los modelos 464 y 472) o bien sobre un disco de 3 pulgadas (en los modelos 664 y 6128). El precio es la ventaja de la cinta sobre el disco pero éste es mucho más rápido y fiable.

Ahora describiremos las instrucciones que hay que usar para grabar o cargar programas tanto en cinta como en disco.

GRABACION Y CARGA DE PROGRAMAS EN CASETE

Si tenemos un programa en la memoria del ordenador y lo queremos grabar en cinta, emplearemos la instrucción:

– SAVE (salvar)

Así, con:

```
SAVE "programa 1"
```

lograremos almacenar en la cinta el programa asignándole un nombre para su posterior identificación (en nuestro caso programa 1).

Si colocamos en la grabadora la cinta casete y tecleamos la anterior instrucción, aparecerá en la pantalla el mensaje:

```
Press REC and PLAY then any key
```

por lo que presionando REC y PLAY del magnetofón y una tecla cualquiera del ordenador se procederá a la grabación en bloques de nuestro programa. En cada momento nos indicará con el mensaje:

```
Saving PROGRAMA 1 block n
```

el bloque que está grabando. Al acabar aparece el mensaje "Ready".

El nombre del programa puede tener desde 0 (cadena vacía) hasta 16 caracteres. Si pretendemos asignarle un nombre con más de 16 caracteres, el ordenador sólo tomará los 16 primeros.

Todos los mensajes que hemos dicho que aparecen en este proceso, excepto "Ready", se pueden evitar con:

```
SAVE "!programa 1"
```

o sea, precediendo el nombre del programa con el signo "!".

Se puede guardar un programa "protegido" con:

```
SAVE "nombre",P
```

con lo que cuando se cargue y ejecute se borrará de la memoria impidiéndose, de esta manera, listarlo.

Tenemos otra posibilidad con:

```
SAVE "nombre",B,m,n
```

que almacena el contenido de las n celdillas de memoria contando a partir de la celdilla número m en adelante.

Como la imagen de la pantalla se guarda a partir de la dirección 49152 y ocupando 16384 celdillas, con:

```
SAVE "imagen",B,49152,16384
```

conseguiremos guardar dicha imagen.

- SPEED WRITE (velocidad de escritura)

Esta instrucción nos permite elegir entre dos velocidades de grabación:

Con:

SPEED WRITE 0

se elige la grabación a velocidad lenta y con:

SPEED WRITE 1

se elige la grabación a velocidad rápida.

La primera es la elegida, si no se indica lo contrario, siendo más fiable que la velocidad rápida, sobre todo con cintas de baja calidad.

- LOAD (cargar)

Para ejecutar la operación inversa, o sea, cargar un programa en el ordenador previamente grabado, emplearemos la instrucción:

LOAD "nombre"

donde se pone entre comillas el nombre del programa que pretendemos cargar. Si se coloca la cadena vacía entre las comillas, se cargará el primer programa que encuentre.

Después de esta instrucción aparece el mensaje:

Press PLAY then any key

con lo que presionando PLAY del casete y luego cualquier tecla del ordenador, aparecerán mensajes del tipo:

Found TITULO block n

que nos indican los diferentes programas que el ordenador va encontrando al leer la cinta. Con el mensaje:

Loading NOMBRE block n

se nos indica que ya ha encontrado el programa y lo está cargando. Al final el mensaje "Ready".

Con LOAD "" cargará el primer programa que encuentre en la cinta.

Con LOAD "!nombre" se suprimen todos los mensajes menos el último.

Es posible, además, suprimir las segundas comillas.

– CAT (CATálogo)

Para conocer los títulos de los programas grabados emplearemos la instrucción:

CAT

después de la cual aparece el conocido mensaje:

Press PLAY then any key

de tal manera que, una vez obedecido, irán apareciendo todos los títulos de los programas grabados.

– RUN (ejecutar)

Hay dos formas de usar esta instrucción:

1) Si tenemos un programa en la memoria y lo queremos ejecutar, emplearemos RUN.

2) Si esta instrucción va seguida de comillas:

   RUN "nombre"

   tiene una misión muy parecida a LOAD pero, además de cargar un programa en la memoria, también lo ejecuta.

   Los programas grabados con protección se cargan con esta instrucción y no con LOAD.

- MERGE (fundir)

Con:

MERGE "prueba"

se mezcla el programa presente hasta ese momento en la memoria con el programa "prueba" leído de la cinta.

Si existen líneas con igual numeración en ambos programas, el programa cargado del casete borrará las líneas del programa primitivo de la misma numeración.

Con esta instrucción no se pueden cargar los programas protegidos.

- CHAIN (encadenar)

Esta instrucción puedė adoptar varias formas que veremos a continuación:

Con:

CHAIN "prueba"

se carga en la memoria el programa "prueba" pero sin borrar las variables definidas con anterioridad.

Con:

CHAIN MERGE "prueba"

obtenemos el mismo efecto que con MERGE pero sin olvidar las variables.

Con:

CHAIN MERGE "prueba",n

además se ejecuta el programa resultante a partir de la línea n.

Con:

CHAIN MERGE "prueba" DELETE m-n

se borran las líneas del programa inicialmente presente en la memoria desde la línea m hasta la n.

UNIDAD DE DISCOS

Aunque las instrucciones para carga y grabación de programas en disco se parecen mucho a las empleadas hasta ahora con cinta, repetiremos su descripción para dejar constancia de ciertas particularidades importantes y también para aquellos lectores que sólo manejan disco.

Una operación previa necesaria antes de proceder a la grabación de programas en disco es el FORMATEADO de éste para poder ser usado más adelante. Esta operación de formateado se ejecuta usando una utilidad en CP/M que se suministra con el ordenador y que se llama FORMAT en el modelo 664 y DISCKIT3 en el modelo 6128.

Para formatear un disco procederemos así:

1) Conectar el ordenador.
2) Introducir el disco CP/M en el modelo 664 o el disco CP/M PLUS en el modelo 6128.
3) Teclear: |CPM
4) En el modelo 664 escribir FORMAT seguido de "ENTER." En el modelo 6128 escribir DISCKIT3 seguido de "INTRO," pulsando la tecla del teclado numérico f4 y después f6 o f9 según el tipo de formateado que interese.

Concluido el formateado de una cara, se procederá a repetir el mismo proceso con la segunda cara del disco.

Es importante advertir que al formatear un disco, toda la información grabada con anterioridad en él se pierde.

El disco lleva un dispositivo, parecido al de una casete, mediante el cual se le protege contra un borrado accidental.

Entremos de lleno en la descripción de las instrucciones de grabación y carga de disco.

– SAVE (salvar)

Con esta instrucción se logra guardar en disco el programa presente en este momento en la memoria del ordenador. Además se le asigna un nombre, en nuestro caso "programa1", que no podrá ser

la cadena vacía, ni tener más de 8 caracteres. Algunos de los caracteres especiales como comas o espacios en blanco tampoco se admiten.

Hay otras maneras de usar esta instrucción que ahora veremos:

SAVE "programa1",P

Hace que se guarde el programa protegido, de tal manera que no se pueda listar.

SAVE "binario",B,m,n

Guarda en disco el contenido de las n celdillas de memoria empezando a contar a partir de la m. Por ejemplo, con la instrucción

SAVE "pantalla",B,49152,16384

se graba la imagen en la pantalla ya que es a partir de la celdilla 49152, y con una longitud de 16384 celdillas, en donde se almacena dicha imagen.

– CAT (CATálogo)

Con esta instrucción conseguimos sacar el catálogo de los programas grabados en una determinada cara del disco. Además dicho catálogo es mostrado mucho más rápidamente que con cinta.

– LOAD (cargar)

Tiene la forma general:

LOAD "programa1"

Carga en la memoria del ordenador, borrando lo almacenado con anterioridad, el programa cuya denominación va entre comillas y que previamente se había grabado en ese disco.

Si un programa se ha grabado protegido, no se podrá cargar con esta instrucción.

- MERGE (fundir)

De forma general:

MERGE "programa2"

y su misión es cargar el programa cuya denominación viene entre comillas mezclándolo con el que hasta ese momento ocupaba la memoria. En el caso de que ambos programas tuviesen líneas con igual numeración, el programa cargado del disco borrará las líneas correspondientes del programa primitivo.

Con esta instrucción tampoco se pueden cargar los programas protegidos.

- RUN (ejecutar)

Tiene dos versiones:

La primera de ellas, RUN, hace que un programa almacenado en la memoria del ordenador se ejecute. Dicho programa ha podido ser tecleado o cargado con LOAD, MERGE,..

La segunda posibilidad:

RUN "programa1"

carga del disco y ejecuta el programa "programa1". Los programas que se han grabado protegidos tienen que ser cargados y ejecutados con esta instrucción.

- CHAIN (encadenar)

Con:

CHAIN "programa2"

se carga el programa cuyo nombre se cita entre comillas pero sin borrar las variables definidas con anterioridad.

Existen otras maneras de usar esta instrucción:

```
CHAIN "programa2",n
```

que además ejecuta el programa a partir de la línea n.

```
CHAIN MERGE "programa2"
```

que tiene el mismo efecto que MERGE pero sin olvidar las variables.

```
CHAIN MERGE "programa2",n
```

lo mismo que en el caso anterior pero además ejecuta el programa resultante a partir de la línea n.

```
CHAIN MERGE "programa2" DELETE m-n
```

tiene el mismo efecto que CHAIN MERGE pero borrando desde la línea m hasta la línea n del programa inicialmente presente en la memoria del ordenador.

Otros libros de la colección GUIA FACIL publicados por

PARANINFO S.A.

ANGULO – DEL MORAL.– Guía fácil. INTELIGENCIA ARTIFICIAL. 104 páginas. 66 figuras.

Presenta el tema al lector interesado que se acerca por primera vez a él. Ofrece una perspectiva general de los apartados más significativos de la Inteligencia Artificial y, sin profundizar en ellos, muestra las bases, las herramientas teóricas y las aplicaciones prácticas.

ANGULO – NO.– Guía fácil. ROBOTICA. 134 páginas. 80 figs.

Introduce el tema de la Robótica de una forma clara y sencilla, evitando demostraciones matemáticas y tecnicismos complejos. Al final de cada capítulo se ha incluido una selecta bibliografía, que ayuda a los interesados a ampliar conocimientos.

BASELGA LOPEZ.– Guía fácil de computadores. ¿Qué equipo necesita usted?

Guía eminentemente práctica que le ayudará a la hora de elegir y comprar un computador que se adapte a sus necesidades.

GHOCHE.– Guía fácil d'BASE III.

Destinado principalmente al aficionado que se proponga ampliar sus conocimientos del d'BASE III o sus equivalentes.

HARTNELL.– Guía fácil de programación de computadores.

Le enseñará a programar rápida y fácilmente. Utilizando términos claros y sencillos, le introducirá en el mundo de la informática y de la programación.

HOLLERBACH.– Guía fácil. PROCESO DE TEXTOS. 104 páginas. 10 figuras.

Esta guía le explica cómo trabaja el procesador y cómo puede serle útil en la oficina.

- Analiza las distintas piezas del hardware.
- Explica cómo se usa un procesador, desde la introducción del texto hasta la manipulación más sofisticada.
- Enseña a evaluar cada sistema de acuerdo con sus necesidades y a superar todos los posibles problemas en la puesta en funcionamiento de su nuevo sistema.

RAMON – BUERA – TRIGO.– Guía fácil MS-DOS

Familiariza al lector con el sistema operativo MS-DOS. Estudia las posibilidades que ofrece este sistema.

SINCLAIR.– Guía fácil. SUBRUTINAS UTILES EN BASIC. 100 páginas.

Contiene listados, notas aclaratorias y listas de variables para muchas subrutinas, entre las que se incluyen, intermitencia del título, impresión en columnas, visualización gráfica enmarcada, título desplazable, subrayado, clasificación, etc.

WATTS.– Guía fácil. COMPUTACION INTERACTIVA. Sistemas. Diálogos. Menús. 96 páginas.

Se analizan las características de los usuarios inexpertos en sistemas interactivos, así como la forma de desarrollar los sistemas apropiados. Se presta mucha atención a los distintos estilos de diálogo: los basados en menús, los de preguntas y respuestas, los basados en órdenes, la consulta, etc.

GUIA FACIL DEL AMSTRAD

El título de esta obra es quizá el argumento más elocuente sobre su contenido. Sus autores han planificado desde las primeras páginas una guía sumamente fácil, apoyando sus descripciones, muy claras y sencillas, con ejemplos, gráficos constantes, que facilitan asimilar rápidamente los conocimientos del lenguaje BASIC en su ordenador AMSTRAD.

Las etapas se suceden dentro de un proceso de enseñanza eminentemente práctico: los programas, ejercicios y ejemplos son desarrollados de manera que cualquier persona, incluso sin ideas previas del lenguaje BASIC, pueda seguirlos no solamente sin esfuerzo alguno, sino amenamente y despertando su interés para ir introduciéndose paulatinamente en temas más complejos.

Puede afirmarse que pocas veces se habrá enfrentado el poseedor de un ordenador AMSTRAD con una obra tan sencillamente expuesta, que va a permitirle adentrarse en las grandes posibilidades que desde ahora va a ofrecerle esta GUIA FACIL para el manejo y explotación del AMSTRAD.

EDITORIAL PARANINFO S.A.
Magallanes, 25 - 28015 Madrid

ISBN: 84-283-1519-1

PARANINFO S.A.

Guía fácil: **AMSTRAD**

Ramón/Buera/Trigo